中国大恒（集团）有限公司北京图像视觉技术分公司

水星系列数字摄像机

快速开发说明书

版本：1.0.0

2012/8/17

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。

中国大恒(集团)有限公司北京图像视觉技术分公司保留对任何产品及相关文件进行修改或改进的权利。本手册的内容若有任何修改，恕不另行通知。

网站：http://www.daheng-image.com

销售信箱：sales@daheng-image.com

销售热线：010-82828878-8025

支持信箱：support@daheng-image.com

支持热线：010-82828878-8006

目录

1. 安装

1.1. 安装步骤

点击 MER_USB_Setup32cn.exe，进入如图 1　1 所示界面；

图 1　1 Step1

点击下一步，进入如图 1　2 所示界面；

图 1　2 Step2

选择我接受协议，点击下一步；

图 1　3 Step3

默认的路径为 C:\Program Files\Daheng Imavision\MER-USBDevice，如果用户要修改安装路径，点击浏览，选择新的路径。然后点击下一步；

图 1　4 Step4

默认选择安装 SDK 和快速软件说明书，点击下一步；

图 1　5 Step5

点击下一步；

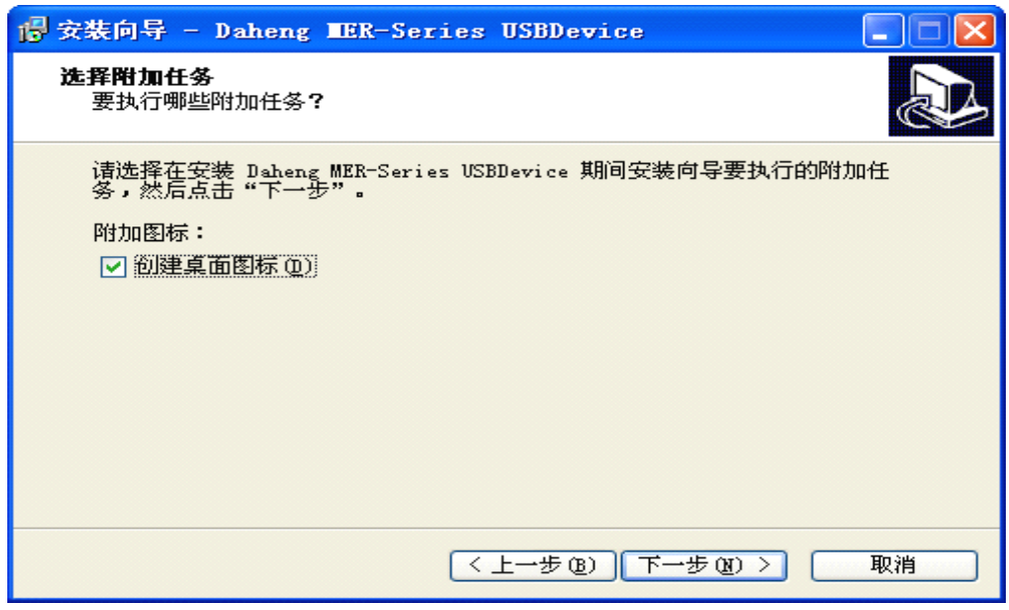


图 1　6 Step6

点击下一步；

图 1　7 Step7

点击安装。在安装过程中，弹出如下对话框；

图 1　8 Step8

选择仍然继续；

安装完成。

图 1　9 Step9

1.2. 说明

安装成功后，SDK 示例程序在安装目录文件夹下的 SDK 文件夹中，帮助文档在 DOC 文件夹中。默认的安装路径为 C:\Program Files\Daheng Imavision\MER-USBDevice，SDK 范例程序在 C:\Program Files\Daheng Imavision\MER-USBDevice\SDK\VC\src 下。

在安装水星摄像机驱动前，如果用户已经安装了大恒图像其它的摄像机驱动，水星摄相机的驱动会更新以前的驱动。所以如果用户要在同一台电脑上同时使用水星 USB 摄像机和其它大恒图像的摄像机，更换大恒不同型号的摄相机前，请重新安装驱动程序。

2.配置环境

在 Microsoft 的32位 Windows 操作系统中，用户可以通过接口库操作控制水星 USB 摄像机。用户在编制自己的应用程序时，可以直接调用这些库函数来实现图像采集和参数设置等功能。

2.1.接口

2.1.1.主功能接口

包括打开摄像机、采集图像、控制摄像机等接口。

包含文件：GxIAPI.h

动态链接库：GxIAPI.dll

静态链接库：GxIAPI.lib

2.1.2.图像处理接口

包括 Bayer 数据转换等接口

包含文件：DxImageProc.h

动态链接库：DxImageProc.dll

静态链接库：DxImageProc.lib

2.2.开发工具

应用接口库支持 32 位编程开发工具 Microsoft Visual C/C++。使用 C/C++编程工具，用户应在程序中调用相关的包含文件（.h），并将静态链接库（.lib）文件加入到工程文件中，供编译程序在链接（Link）时使用。

2.3.用户工程创建

2.3.1.工程配置(Viscual C++6.0)

在 project->settings 的 link 选项卡中，Category 下拉框选择 Gengral 选项，在 Object/library modules 下加入 GxIAPI.lib DxImageProc.lib，如图 2 1 所示。另外在用户的工程中添加 GxIAPI.h 和 DxImageProc.h 头文件。

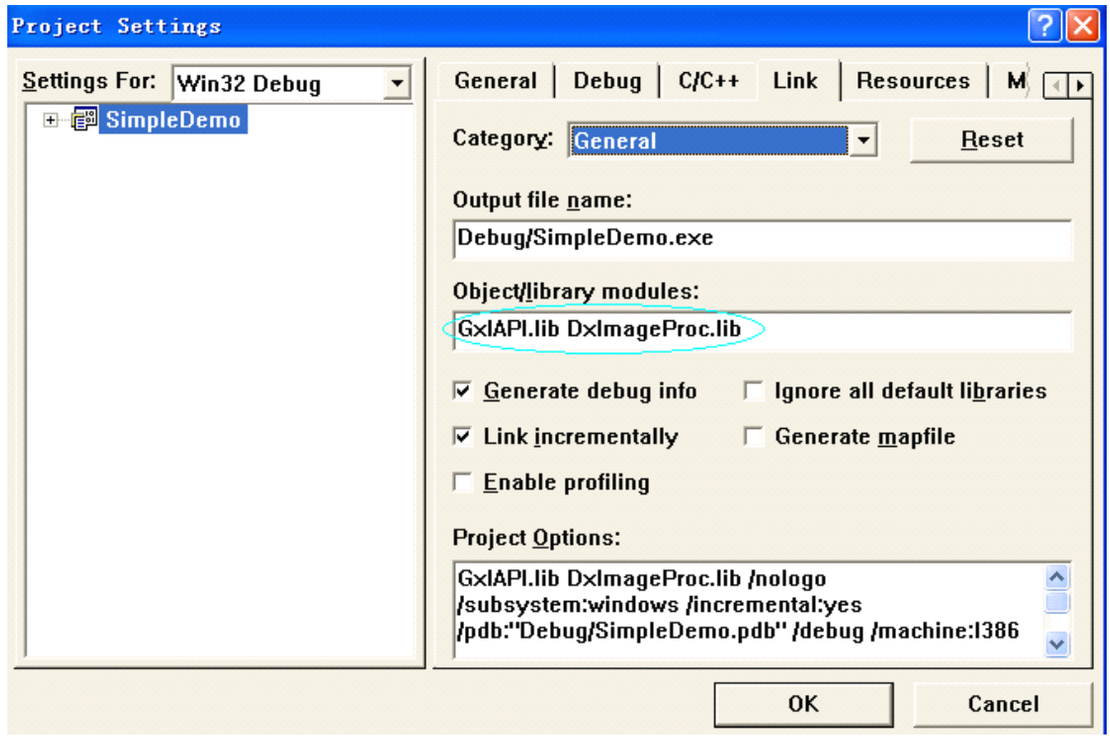


图 2 1 添加静态库（VC6.0）

3. 如何采集图像

3.1. 图像采集流程图

图像采集流程图如下：

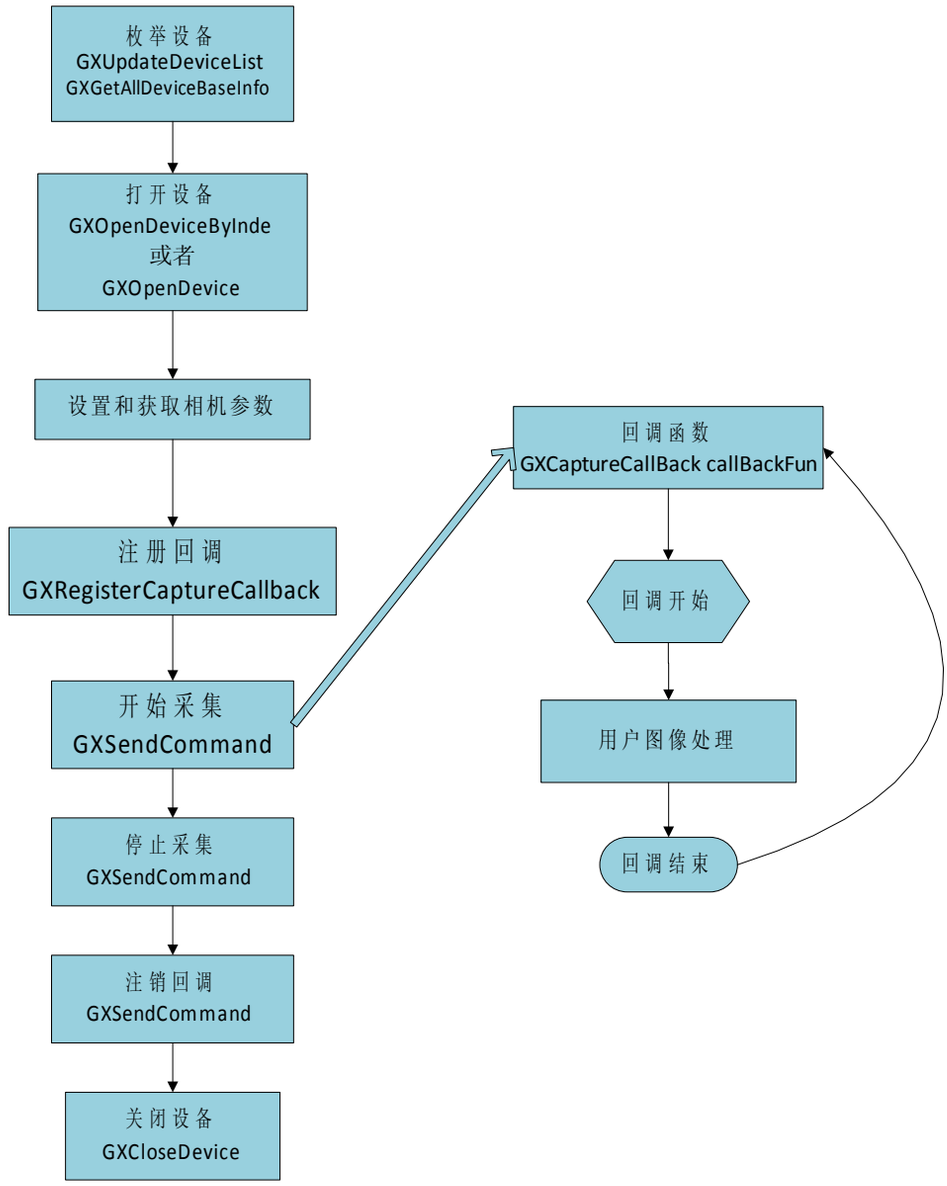


图 3 1 图像采集流程图

采集图像流程代码举例：

```
#include "stdafx.h"
#include "GxIAPI.h"
#include "iostream.h"
#define    IS_SNAP_SINGLE 0
BYTE *pImageBuffer;///采集图像数据缓冲区
static void __stdcall OnFrameCallbackFun(GX_FRAME_CALLBACK_PARAM* pFrame)
{
      //可对 pImageBuffer 进行图像处理或者显示操作
      //……
      //结束图像处理操作
}
int main(int argc, char* argv[])
{
      GX_DEV_HANDLE    m_hDevice;; //USB 数字摄像机句柄
      GX_STATUS status = GX_STATUS_SUCCESS;
      pImageBuffer = new BYTE[2048 * 1536 ];
      GXInitLib();      //初始化库
      uint32_t m_nNumberDevice;
      GXUpdateDeviceList(&m_nNumberDevice,1000);//获得设备个数
      GX_DEVICE_BASE_INFO *baseinfo = new GX_DEVICE_BASE_INFO[m_nNumberDevice];
      size_t nSize = m_nNumberDevice * sizeof(GX_DEVICE_BASE_INFO);//获取设备信息
      status = GXGetAllDeviceBaseInfo(baseinfo, &nSize);
      status =GXOpenDeviceByIndex(1, &m_hDevice);; //打开数字相机，注：已经包含默认参数
设
      status =GXSetInt(m_hDevice,GX_INT_WIDTH,2048);//设置图像宽度 2048
      status =GXSetInt(m_hDevice,GX_INT_HEIGHT,1536);//设置图像高度 1536
      status
=GXSetEnum(m_hDevice,GX_ENUM_ACQUISITION_MODE,GX_ACQ_MODE_CONTINUOUS);//设
置采集模式为连续采集
      status =GXSetInt(m_hDevice,GX_INT_ACQUISITION_SPEED_LEVEL,8);//设置采集速度，范围
(0-12)
      status =GXSetInt(m_hDevice,GX_INT_GAIN,8);//设置增益,增益范围(0-63)
      status =GXSetFloat(m_hDevice,GX_FLOAT_EXPOSURE_TIME,30000);//曝光时间 30ms
#if IS_SNAP_SINGLE      //采集单帧模式
      GX_FRAME_DATA frameData;
      frameData.pImgBuf = pImageBuffer;
      frameData.nStatus = -1;
      status = GXSendCommand(m_hDevice, GX_COMMAND_ACQUISITION_START);
      do
      {
            status = GXGetImage(m_hDevice, &frameData, 20);
      } while(frameData.nStatus != 0);
```

```
    //可对 pImageBuffer 进行图像处理或者显示操作
    //      ……
    //结束图像处理操作
    status = GXSendCommand(m_hDevice, GX_COMMAND_ACQUISITION_STOP);
#else//连续采集模式

    status = GXRegisterCaptureCallback(m_hDevice,pImageBuffer, OnFrameCallbackFun);//注
册回调函数
    status = GXSendCommand(m_hDevice, GX_COMMAND_ACQUISITION_START);
    Sleep(2000);//延迟 2s，等待回调函数采集 n 帧图像
    status = GXSendCommand(m_hDevice, GX_COMMAND_ACQUISITION_STOP);
    status = GXUnregisterCaptureCallback(m_hDevice);//注销回调函数

#endif

    status = GXCloseDevice(m_hDevice);
    delete []pImageBuffer;//回收图像缓冲区
    GXCloseLib();              //关闭库
    return 0;
}
```

3.2.采集图像步骤说明

3.2.1.枚举设备

- 获取设备个数

函数：GXUpdateDeviceList
原型：GXUpdateDeviceList (uint32_t* punNumDevices, int32_t unTimeOut);
参数：punNumDevices;　　指向设备个数的整型指针
　　unTimeOut;　　超时时间
使用接口 GXUpdateDeviceList 可获取设备个数，范例如下：

```
GX_STATUS status = GX_STATUS_SUCCESS;
uint32_t nDeviceNum = 0;
int32_t    untimeout=1000;
status = GXUpdateDeviceList(&nDeviceNum, untimeout);
```

- 获取设备信息

函数：GXGetAllDeviceBaseInfo
原型：GXGetAllDeviceBaseInfo (GX_DEVICE_BASE_INFO* pDeviceInfo,
　　size_t* pBufferSize);
参数：
　　pDeviceInfo；　　指向设备基本信息的结构体指针
　　pBufferSize；　　指向设备信息所需 Buffer 大小的指针
设备基本信息结构体定义：

```
typedef struct GX_DEVICE_BASE_INFO
{
    char szVendorName[32];          ///< 厂商名称
    char szModelName[32];           ///< 设备类型名称
    char szSN[32];                  ///< 设备序列号
    char szDisplayName[64];         ///< 设备展示名称
    char reserved[512];             ///< 保留占位
}GX_DEVICE_BASE_INFO;
```

使用接口 GXGetAllDeviceBaseInfo 获取所有设备信息，范例如下：

```
GX_STATUS status = GX_STATUS_SUCCESS;
GX_DEVICE_BASE_INFO *baseinfo
        = newGX_DEVICE_BASE_INFO[nDeviceNum];// nDeviceNum 为设备数目
size_t nSize = nDeviceNum * sizeof(GX_DEVICE_BASE_INFO);
status = GXGetAllDeviceBaseInfo(baseinfo, &nSize);
```

3.2.2.打开设备

函数：GXOpenDeviceByIndex
函数原型：GXOpenDeviceByIndex(uint32_t　nDeviceIndex,
　　GX_DEV_HANDLE* phDevice)
参数：
　　nDeviceIndex;　　设备序号(从 1 开始)
　　phDevice;　　指向设备句柄的指针
使用接口 GXOpenDeviceByIndex 可按顺序打开设备，范例如下：

```
GX_STATUS status = GX_STATUS_SUCCESS;
status = GXOpenDeviceByIndex(1, &m_hDevice);
```

另外，还可以用 GXOpenDevice 打开设备。

3.2.3.设置及获取摄像机参数

详见第四章如何设置和获取摄像机参数。

3.2.4.注册回调

函数：GXRegisterCaptureCallback

函数原型：GXRegisterCaptureCallback (GX_DEV_HANDLE hDevice,
void *pFrameData,
GXCaptureCallBack callBackFun);

参数:

hDevice; 设备句柄

pFrameData; 用户自定义数据指针

callbackFun; 回调函数

使用接口 GXRegisterCaptureCallback 注册回调，范例如下：

```
GX_STATUS status = GX_STATUS_SUCCESS;
status = GXRegisterCaptureCallback(m_hDevice, this, OnFrameCallbackFun);
```

3.2.5.开始采集

使用接口 GXSendCommand 调用开始采集命令。

函数：GXSendCommand

函数原型：GXSendCommand(GX_DEV_HANDLE hDevice,
GX_FEATURE_ID featureID);

参数：

hDevice：设备句柄

featureID；Command 型宏

函数功能：发送命令

范例如下：

```
GX_STATUS status = GX_STATUS_SUCCESS;
status = GXSendCommand(m_hDevice, GX_COMMAND_ACQUISITION_START);
```

3.2.6.开始采集之后注意事项

开始采集之后为了保证出图的稳定性有一些功能会被锁定，锁定状态下的功能不能够进行读写控制。这些功能在停止采集之后恢复正常读写控制。

开始采集之后会被锁定的功能有：

GX_INT_OFFSET_X

GX_INT_OFFSET_Y

GX_INT_WIDTH

GX_INT_HEIGHT

GX_INT_BINNING_HORIZONTAL

GX_INT_BINNING_VERTICAL

GX_INT_DECIMATION_HORIZONTAL

GX_INT_DECIMATION_VERTICAL

GX_ENUM_ACQUISITION_MODE

GX_INT_ACQUISITION_SPEED_LEVEL

GX_ENUM_TRIGGER_MODE

GX_COMMAND_USER_SET_LOAD

3.2.7.停止采集

使用接口 GXSendCommand 调用停止采集命令，范例如下：

```
GX_STATUS status = GX_STATUS_SUCCESS;
status = GXSendCommand(m_hDevice, GX_COMMAND_ACQUISITION_STOP);
```

3.2.8.注销回调

函数：GXUnregisterCaptureCallback

函数原型：GXUnregisterCaptureCallback(GX_DEV_HANDLE hDevice);

参数：hDevice;　设备句柄

使用接口 GXUnregisterCaptureCallback 取消注册回调，范例如下：

```
GX_STATUS status = GX_STATUS_SUCCESS;
status = GXUnregisterCaptureCallback(m_hDevice);
```

3.2.9.关闭设备

函数：GXCloseDevice

函数原型：GXCloseDevice(GX_DEV_HANDLE hDevice);

参数：hDevice；设备句柄

函数使用举例：

```
GX_STATUS status = GX_STATUS_SUCCESS;
GXCloseDevice(m_hDevice);
```

3.2.10. 回调函数

假设注册回调时函数命名为 OnFrameCallbackFun。则回调函数的形式为：

void __stdcall OnFrameCallbackFun(GX_FRAME_CALLBACK_PARAM* pFrame);

GX_FRAME_CALLBACK_PARAM 结构体定义：

```
typedef struct GX_FRAME_CALLBACK_PARAM
{
  void*          pUserParam;          ///< 用户私有数据
  int32_t        status;              ///< 获取该帧图像的返回状态
  const  void*   pImgBuf;             ///< 返回的图像 buffer 地址
  int32_t        nImgSize;            ///< 返回的图像大小
  int32_t        reserved[8];         ///< 保留
}GX_FRAME_CALLBACK_PARAM;
```

4.如何设置和获取摄像机参数

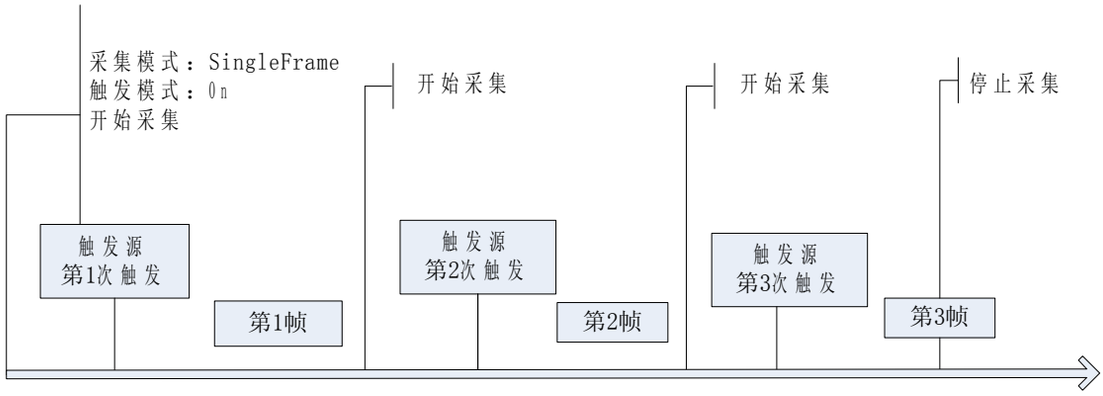
4.1.采集模式和触发模式

4.1.1.水星系列摄像机采集模式说明

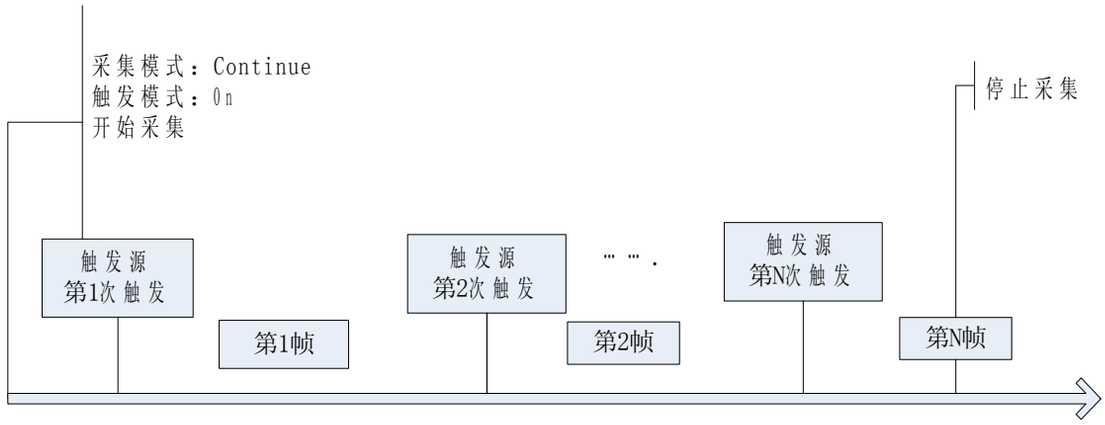
水星系列摄像机的采集图像方式有采集模式和触发模式共同控制，如表 4 1所示。

采集模式	触发模式	组合模式
SingleFrame	on	Model1(触发采单帧)
continue	on	Model2（触发连续采集）
SingleFrame	off	Model3（非触发采单帧）
continue	off	Model4（非触发连续采集）

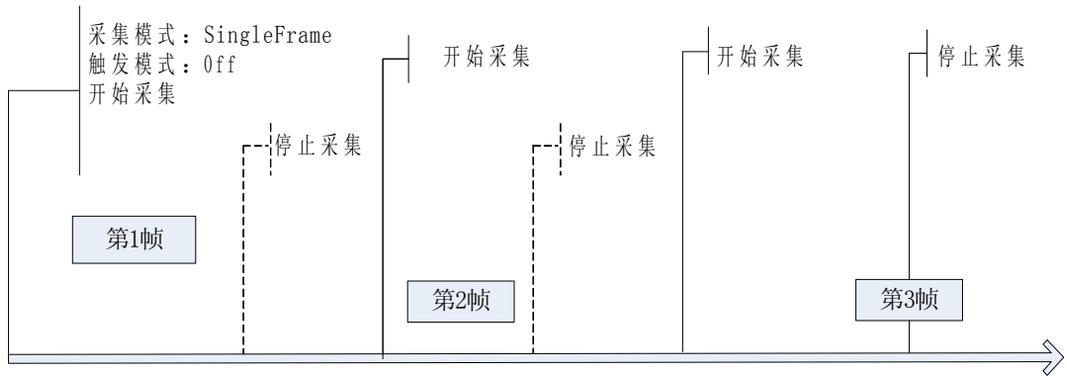
表 4 1 水星系列摄像机采集模式



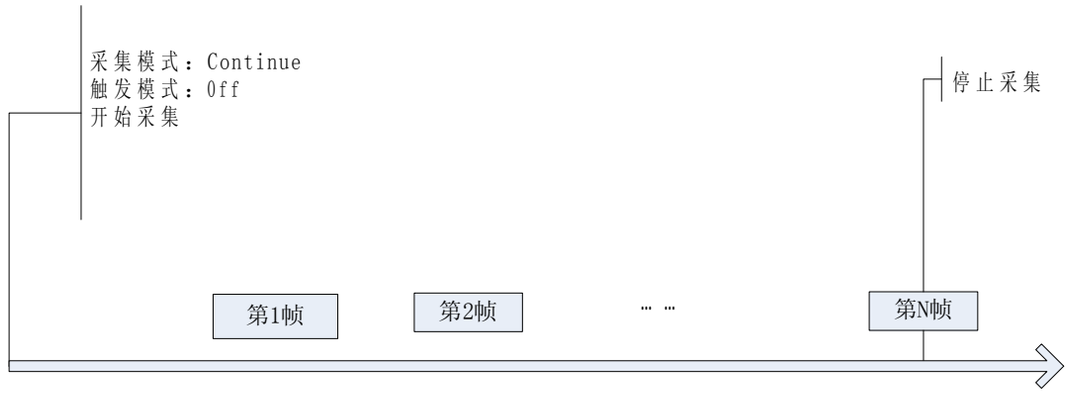
Model1 模式（触发采单帧）



Model2 模式（触发连续采集）



Model3 模式（非触发采单帧）



Model4 模式（非触发连续采集）

4.1.2.采集模式

4.1.2.1. 设置采集模式

设置采集模式使用 GXSetEnum 接口，设置采集模式需在摄像机停止采集状态下才能设置。
函数: GXSetEnum

函数原型：GXSetEnum(GX_DEV_HANDLE hDevice,

GX_FEATURE_ID featureID,
int64_t pnValue)

参数：

hDevice： 设备句柄
featureID： 枚举型参数宏
pnValue： 需设置的枚举型参数的值

函数功能：设置枚举型摄像机参数值

枚举型参数宏 GX_ENUM_ACQUISITION_MODE 对应的枚举型定义为：

```
typedef enum GX_ACQUISITION_MODE_ENTRY
{
    GX_ACQ_MODE_SINGLE_FRAME = 0,                    ///<单帧模式
    GX_ACQ_MODE_CONTINUOUS   = 2,                    ///<连续模式
}GX_ACQUISITION_MODE_ENTRY;
```

程序代码举例：

```
GX_STATUS status = GX_STATUS_SUCCESS;
status = GXSetEnum(m_hDevice, GX_ENUM_ACQUISITION_MODE,
                   GX_ACQ_MODE_CONTINUOUS);
```

4.1.2.2. 获取采集模式

设置采集模式使用 GXGetEnum 和接口。

函数: GXGetEnum

函数原型：GXGetEnum(GX_DEV_HANDLE hDevice,
GX_FEATURE_ID featureID,
int64_t* pnValue)

参数：

hDevice： 设备句柄
featureID： 枚举型参数宏
pnValue： 指向枚举型参数的整型指针

函数功能：获取枚举型摄像机参数值

程序代码举例：

```
GX_STATUS status = GX_STATUS_SUCCESS;
int64_t nAcquisitionMode ;
status = GXGetEnum(m_hDevice, GX_ENUM_ACQUISITION_MODE,
                   &nAcquisitionMode);
```

4.1.2.3. 获取采集模式描述

设置采集模式使用 GXGetEnumDescription 和接口。

函数：GXGetEnumDescription

函数原型：GXGetEnumDescription (GX_DEV_HANDLE hDevice,
GX_FEATURE_ID featureID,

GX_ENUM_DESCRIPTION*pEnumDescription,
size_t* pBufferSize)

参数：

hDevice； 设备句柄

featureID； 枚举型参数宏

pEnumDescription；指向枚举型描述结构体的指针

pBufferSize; 指向 Buffer 大小的整型指针

函数功能：获取枚举型参数描述

程序代码举例：

```
GX_STATUS status = GX_STATUS_SUCCESS;
GX_ENUM_DESCRIPTION *pEnum = NULL;
size_t bufferSize = 0;
status = GXGetEnumDescription(hDevice, GX_ENUM_ACQUISITION_MODE,
        pEnum, &bufferSize);//第 1 次调用获取所需 Buffer 大小
 size_t nEntryNum = bufferSize/sizeof(GX_ENUM_DESCRIPTION);
 pEnum = new GX_ENUM_DESCRIPTION[nEntryNum];
 status = GXGetEnumDescription(m_hDevice, GX_ENUM_ACQUISITION_MODE,
         pEnum, &bufferSize);//第二次获取采集模式枚举描述
 for (uint32_t i=0; i<nEntryNum; i++)
 {
   m_combACQMode.SetItemData(
             m_combACQMode.AddString(pEnum[i].szSymbolic),
             (uint32_t)pEnum[i].nValue);//m_combACQMode 为 CComboBox 类
 }
 delete []pEnum;
```

获取枚举描述时需调用两次，第一次调用获得枚举型描述所需 Bufffer 大小，第二次调用获取枚举描述。

4.1.3.触发模式

触发模式为枚举型参数，调用 GXGetEnum、GXSetEnum 和 GXGetEnumDescription 接口。

4.1.3.1. 设置触发模式

程序举例：开启触发模式

```
GX_STATUS status = GX_STATUS_SUCCESS;
status = GXSetEnum(m_hDevice, GX_ENUM_TRIGGER_MODE,
                   GX_TRIGGER_MODE_ON);
```

4.1.3.2. 获取触发模式

程序举例：

```
GX_STATUS status = GX_STATUS_SUCCESS;
int64_t nTriggerMode ;
```

```
status = GXGetEnum(m_hDevice, GX_ENUM_TRIGGER_MODE,
                                &nTriggerMode);
```

4.2.曝光时间

4.2.1.设置曝光时间

通过 GXSetFloat 接口设置曝光时间。

函数：GXSetFloat

函数原型：GXSetFloat (GX_DEV_HANDLE hDevice,
                        GX_FEATURE_ID featureID,
                        double dValue);

参数：

hDevice;　设备句柄

featureID;　浮点型参数宏

dValue;　设置值

函数功能：设置浮点型摄像机参数值

程序举例：

```
GX_STATUS status = GX_STATUS_SUCCESS;
double dExporeTime=30000;       //30ms
status = GXSetFloat(m_hDevice, GX_FLOAT_EXPOSURE_TIME, dExporeTime);
```

4.2.2.获取曝光时间

通过 GXGetFloat 接口设置曝光时间。

函数：GXGetFloat

函数原型：GXGetFloat(GX_DEV_HANDLE hDevice,
                        GX_FEATURE_ID featureID,
                        double* pdValue);

参数：

hDevice;　设备句柄

featureID;　浮点型参数宏

dValue;　指向返回浮点型参数的指针

函数功能：获取浮点型摄像机参数值

程序举例：

```
GX_STATUS status = GX_STATUS_SUCCESS;
double dExporeTime;
status = GXGetFloat(m_hDevice, GX_FLOAT_EXPOSURE_TIME, &dExporeTime);
```

4.2.3.获取曝光时间范围

通过 GXGetFloatRange 接口获取曝光时间范围。

函数原型：GXGetFloatRange(GX_DEV_HANDLE hDevice,
GX_FEATURE_ID featureID,
GX_FLOAT_RANGE* pFloatRange);

参数：

hDevice; 设备句柄

featureID； 浮点型参数宏

pFloatRange; 指向 GX_FLOAT_RANGE 的结构体指针

函数功能：获取浮点型参数范围

结构体 GX_FLOAT_RANGE 定义：

```
typedef struct GX_FLOAT_RANGE
{
  double   dMin;                    ///< 浮点型最小值
  double   dMax;                    ///< 浮点型最大值
  double   dInc;                    ///< 浮点型步长
  char     szUnit[8];               ///< 浮点型单位
  int32_t reserved[8];              ///< 保留
}GX_FLOAT_RANGE;
```

程序举例：

```
GX_STATUS status = GX_STATUS_SUCCESS;
GX_FLOAT_RANGE FRange;
status = GXGetFloatRange(m_hDevice, GX_FLOAT_EXPOSURE_TIME,&FRange);
```

4.3.增益

4.3.1.设置增益

通过 GXSetInt 接口设置增益。

函数：GXSetInt

函数原型： GXSetInt(GX_DEV_HANDLE hDevice,
GX_FEATURE_ID featureID,
int64_t pnValue);

参数： hDevice； 设备句柄

featureID； 整型参数宏

pnValue； 需设置的整形参数的值

函数功能：设置摄像机整型参数值

程序代码举例：

```
int64_t nGain=8 ;
GX_STATUS status = GX_STATUS_SUCCESS;
```

```
status = GXSetInt(m_hDevice, GX_INT_GAIN, nGain);
```

4.3.2.获取增益

通过 GXGetInt 获取增益。

函数：GXGetInt

函数原型： GXGetInt(GX_DEV_HANDLE hDevice,
GX_FEATURE_ID featureID,
int64_t* pnValue);

参数：

hDevice； 设备句柄

featureID； 整型参数宏

pnValue； 指向整型参数的指针

函数功能：获取整型摄像机参数

程序代码举例：

```
int64_t nGain ;
GX_STATUS status = GX_STATUS_SUCCESS;
status = GXGetInt(m_hDevice, GX_INT_GAIN, &nGain);
```

4.3.3.获取增益范围

通过 GXGetIntRange 获取增益。

函数：GXGetIntRange

函数原型：GXGetIntRange(GX_DEV_HANDLE hDevice,
GX_FEATURE_ID featureID,
GX_INT_RANGE* pIntRange);

参数：

hDevice； 设备句柄

featureID；整型摄像机参数宏

pIntRange；指向 GX_INT_RANGE 的结构体指针

函数功能：获取整型参数范围

结构体 GX_INT_RANGE 定义：

```
typedef struct GX_INT_RANGE
{
  int64_t nMin;                    ///< 整型值最小值
  int64_t nMax;                    ///< 整型值最大值
  int64_t nInc;                    ///< 整型值步长
  int32_t reserved[8];             ///< 保留
}GX_INT_RANGE;
```

程序代码举例：

```
GX_STATUS status = GX_STATUS_SUCCESS;
GX_INT_RANGE nRange;
status = GXGetIntRange(m_hDevice, GX_INT_GAIN,&nRange);
```

4.4. 采集速度级别

采集速度参数和增益参数都是整型参数，都调用 GXGetInt、GXSetInt 和 GXGetIntRange 接口。设置采集速度需在摄像机停止采集状态下才能设置。

- 设置采集速度级别程序代码：

```
GX_STATUS status = GX_STATUS_SUCCESS;
int64_t nSpeed= 8;
status = GXSetInt(m_hDevice, GX_INT_ACQUISITION_SPEED_LEVEL, nSpeed);
```

- 获取采集速度级别程序代码

```
int64_t nSpeed;
GX_STATUS status = GX_STATUS_SUCCESS;
status = GXGetInt(m_hDevice, GX_INT_ACQUISITION_SPEED_LEVEL, & nSpeed);
```

- 获取采集速度级别范围程序代码

```
GX_STATUS status = GX_STATUS_SUCCESS;
GX_INT_RANGE nRange;
status = GXGetIntRange(m_hDevice, GX_INT_ACQUISITION_SPEED_LEVEL,&nRange);
```

4.5. 图像宽度和图像高度

图像宽度和图像高度为整型参数，调用 GXGetInt、GXSetInt 和 GXGetIntRange 接口。设置图像宽度需在摄像机停止采集状态下才能设置。

- 设置图像宽度程序代码：

```
GX_STATUS status = GX_STATUS_SUCCESS;
int64_t nWidth = 480;
status = GXSetInt(m_hDevice, GX_INT_WIDTH, nWidth);
```

- 获取图像宽度程序代码

```
int64_t nWidth ;
GX_STATUS status = GX_STATUS_SUCCESS;
status = GXGetInt(m_hDevice, GX_INT_WIDTH, &nWidth);
```

- 获取图像宽度范围程序代码

```
GX_STATUS status = GX_STATUS_SUCCESS;
GX_INT_RANGE nRange;
status = GXGetIntRange(m_hDevice, GX_INT_WIDTH,&nRange);
```

- 设置图像高度程序代码：

```
GX_STATUS status = GX_STATUS_SUCCESS;
int64_t nHeight= 480;
status = GXSetInt(m_hDevice, GX_INT_HEIGHT, nHeight);
```

- 获取图像高度程序代码

```
int64_t Height;
```

```
GX_STATUS status = GX_STATUS_SUCCESS;
status = GXGetInt(m_hDevice, GX_INT_HEIGHT, &Height);
```

- 获取图像高度范围程序代码

```
GX_STATUS status = GX_STATUS_SUCCESS;
GX_INT_RANGE nRange;
status = GXGetIntRange(m_hDevice, GX_INT_HEIGHT,&nRange);
```

4.6. 自动白平衡

自动白平衡为枚举型参数，调用 GXGetEnum、GXSetEnum 和 GXGetEnumDescription 接口。

白平衡枚举定义：

```
typedef enum GX_BALANCE_WHITE_AUTO_ENTRY
{
  GX_BALANCE_WHITE_AUTO_OFF = 0,                    ///< 关闭自动白平衡
  GX_BALANCE_WHITE_AUTO_CONTINUOUS = 1,          ///< 连续自动白平衡
  GX_BALANCE_WHITE_AUTO_ONCE = 2,                 / //< 单次自动白平衡
}GX_BALANCE_WHITE_AUTO_ENTRY;
```

程序举例：开启连续自动白平衡

```
GX_STATUS status = GX_STATUS_SUCCESS;
status = GXSetEnum(m_hDevice, GX_ENUM_BALANCE_WHITE_AUTO,
                   GX_BALANCE_WHITE_AUTO_CONTINUOUS);
```

程序举例：开启单次自动白平衡

```
GX_STATUS status = GX_STATUS_SUCCESS;
status = GXSetEnum(m_hDevice, GX_ENUM_BALANCE_WHITE_AUTO,
                     GX_BALANCE_WHITE_AUTO_ONCE);
```

程序举例：查看自动白平衡状态

```
GX_STATUS status = GX_STATUS_SUCCESS;
int64_t nBalance_White ;
status = GXGetEnum(m_hDevice, GX_ENUM_BALANCE_WHITE_AUTO,
                     & nBalance_White);
```

4.7. 自动颜色校正

自动颜色校正为枚举型参数，调用 GXGetEnum、GXSetEnum 和 GXGetEnumDescription 接口。

程序举例：开启白平衡

```
GX_STATUS status = GX_STATUS_SUCCESS;
```

```
status = GXSetEnum(m_hDevice, GX_ENUM_COLOR_CORRECT,
                   GX_COLOR_CORRECT_ON);
```

程序举例：查看自动白平衡是否开启

```
GX_STATUS status = GX_STATUS_SUCCESS;
int64_t nColor_Correct ;
status = GXGetEnum(m_hDevice, GX_ENUM_COLOR_CORRECT,
                   & nColor_Correct);
```

5. 附录

A. 函数表

函数	函数说明
GXInitLib()	初始化库
GXCloseLib()	关闭库
GXUpdateDeviceList (uint32_t* punNumDevices, int32_t unTimeOut)	获取设备个数
GXGetAllDeviceBaseInfo (GX_DEVICE_BASE_INFO* pDeviceInfo, size_t* pBufferSize)	获取摄像机信息
GXOpenDeviceByIndex(uint32_t nDeviceIndex, GX_DEV_HANDLE* phDevice)	按设备号打开摄像机
GXOpenDevice (GX_OPEN_PARAM* pOpenParam, GX_DEV_HANDLE* phDevice)	打开设备
GXCloseDevice (GX_DEV_HANDLE hDevice)	关闭设备
GXIsImplemented(GX_DEV_HANDLE hDevice, GX_FEATURE_ID featureID, bool* pbIsImplemented)	查询是否支持该功能
GXIsReadable(GX_DEV_HANDLE hDevice, GX_FEATURE_ID featureID, bool* pbIsReadable)	查询参数是否可读
GXIsWritable (GX_DEV_HANDLE hDevice, GX_FEATURE_ID featureID, bool* pbIsWritable)	查询参数是否可写
GXGetIntRange(GX_DEV_HANDLE hDevice, GX_FEATURE_ID featureID, GX_INT_RANGE* pIntRange);	获取整型参数范围
GXGetInt (GX_DEV_HANDLE hDevice, GX_FEATURE_ID featureID, int64_t* pnValue)	获取整型参数值
GXSetInt (GX_DEV_HANDLE hDevice, GX_FEATURE_ID featureID, int64_t nValue)	设置整型参数值
GXSetFloat (GX_DEV_HANDLE hDevice, GX_FEATURE_ID featureID, double dValue);	设置浮点型参数值
GXGetFloat (GX_DEV_HANDLE hDevice, GX_FEATURE_ID featureID, double* pdValue)	获取浮点型参数值
GX_API GXGetFloatRange(GX_DEV_HANDLE hDevice, GX_FEATURE_ID featureID, GX_FLOAT_RANGE* pFloatRange);	获取浮点型参数范围
GXGetEnumDescription (GX_DEV_HANDLE hDevice, GX_FEATURE_ID featureID, GX_ENUM_DESCRIPTION*pEnumDescription, size_t* pBufferSize)	获取枚举描述
GXGetEnum(GX_DEV_HANDLE hDevice, GX_FEATURE_ID featureID, int64_t* pnValue)	获取枚举型参数值

GXSetEnum(GX_DEV_HANDLE hDevice, GX_FEATURE_ID featureID, int64_t nValue)	设置枚举型 参数值
GXGetBool(GX_DEV_HANDLE hDevice, GX_FEATURE_ID featureID, bool* pbValue)	获取布尔型 参数值
GXSetBool(GX_DEV_HANDLE hDevice, GX_FEATURE_ID featureID, bool bValue)	设置布尔型 参数值
GXGetStringLength(GX_DEV_HANDLE hDevice, GX_FEATURE_ID featureID, size_t* pnSize)	获取字符串参数长度
GXGetString(GX_DEV_HANDLE hDevice, GX_FEATURE_ID featureID, char* pszContent, size_t* pnSize)	获取字符串
GXSetString(GX_DEV_HANDLE hDevice, GX_FEATURE_ID featureID, char* pszContent)	设置字符串值
GXGetBufferLength(GX_DEV_HANDLE hDevice, GX_FEATURE_ID featureID, size_t* pnSize)	获取 Buffer 型参数长度
GXGetBuffer(GX_DEV_HANDLE hDevice, GX_FEATURE_ID featureID, uint8_t* pBuffer, size_t* pnSize)	获取 Buffer 型
GXSetBuffer (GX_DEV_HANDLE hDevice, GX_FEATURE_ID featureID, uint8_t* pBuffer, size_t nSize)	设置 Buffer 数据
GXSendCommand (GX_DEV_HANDLE hDevice, GX_FEATURE_ID featureID)	发送命令
GXGetFeatureName (GX_DEV_HANDLE hDevice, GX_FEATURE_ID featureID, char* pszName, size_t* pnSize)	获取宏描述
GXRegisterCaptureCallback (GX_DEV_HANDLE hDevice, void *pFrameData, GXCaptureCallBack callBackFun);	注册回调
GXUnregisterCaptureCallback(GX_DEV_HANDLE hDevice);	取消注册回调
GXGetImage (GX_DEV_HANDLE hDevice, GX_FRAME_DATA *pFrameData, int32_t nTimeout)	获取一幅图像 (无回调)

B. 功能表

功能		函数/宏	类型
采集摄像机	枚举摄像机	GXUpdateDeviceList	函数
		GXGetAllDeviceBaseInfo	函数
	打开设备	GXOpenDeviceByIndex	函数

		GXOpenDevice	函数
	关闭设备	GXCloseDevice	函数
	注册回调	GXRegisterCaptureCallback	函数
	取消注册	GXUnregisterCaptureCallback	函数
	开始采集命令	GX_COMMAND_ACQUISITION_START	宏
	停止采集命令	GX_COMMAND_ACQUISITION_STOP	宏
	初始化库	GXInitLib	函数
	关闭库	GXCloseLib	函数
参数设置及获取功能	序列号	GX_STRING_DEVICE_ID	宏
	图像宽度	GX_INT_WIDTH	宏
	图像高度	GX_INT_HEIGHT	宏
	数据格式	GX_ENUM_PIXEL_FORMAT	宏
	X 方向偏移	GX_INT_OFFSET_X	宏
	Y 方向偏移	GX_INT_OFFSET_Y	宏
	采集模式	GX_ENUM_ACQUISITION_MODE	宏
	采集速度级别	GX_INT_ACQUISITION_SPEED_LEVEL	宏
	采多帧模式下的帧数	GX_INT_ACQUISITION_FRAME_COUNT	宏
	触发模式	GX_ENUM_TRIGGER_MODE	宏
	软触发命令	GX_COMMAND_TRIGGER_SOFTWARE	宏
	触发极性	GX_ENUM_TRIGGER_ACTIVATION	宏
	触发开关	GX_ENUM_TRIGGER_SWITCH	宏
	曝光时间	GX_FLOAT_EXPOSURE_TIME	宏
	自动曝光开关	GX_ENUM_EXPOSURE_AUTO	宏
	闪光灯模式开关	GX_ENUM_STROBE_SWITCH	宏
	自动增益开关	GX_ENUM_GAIN_AUTO	宏
	增益	GX_INT_GAIN	宏
	自动增益最小值	GX_INT_AUTO_GAIN_VALUEMIN	宏
	自动增益最大值	GX_INT_AUTO_GAIN_VALUEMAX	宏
	自动曝光最小值	GX_INT_AUTO_SHUTTER_VALUEMIN	宏
	自动曝光最大值	GX_INT_AUTO_SHUTTER_VALUEMAX	宏
	自动黑电平开关	GX_ENUM_BLACKLEVEL_AUTO	宏
	全局黑电平	GX_INT_BLACKLEVEL	宏
	Bayer 格式	GX_ENUM_PIXEL_COLOR_FILTER	宏
	增益红通道	GX_INT_GAIN_RED	宏
	增益绿 1 通道	GX_INT_GAIN_GREEN1	宏
	增益绿 2 通道	GX_INT_GAIN_GREEN2	宏
	增益蓝通道	GX_INT_GAIN_BLUE	宏
	黑电平绿 1 通道	GX_INT_BLACKLEVEL_GREEN1	宏
	黑电平绿 2 通道	GX_INT_BLACKLEVEL_GREEN2	宏
	黑电平蓝通道	GX_INT_BLACKLEVEL_BLUE	宏
	自动白平衡开关	GX_ENUM_BALANCE_WHITE_AUTO	宏

	白平衡系数红通道	GX_FLOAT_BALANCE_RATIO_RED	宏
	白平衡系数绿 1 通道	GX_FLOAT_BALANCE_RATIO_GREEN1	宏
	白平衡系数绿 2 通道	GX_FLOAT_BALANCE_RATIO_GREEN2	宏
	白平衡系数蓝通道	GX_FLOAT_BALANCE_RATIO_BLUE	宏
	颜色校正开关	GX_ENUM_COLOR_CORRECT	宏