# 访问存储区域

本手册描述具有扩展存储空间的工业相机（-2）的存储空间的访问方法。

## 1 存储结构

相机使用 SPI 存储程序和数据，标准的相机型号使用的存储芯片是 32Mbit（4MB）的 Flash。 存放有两个启动固件以及 3 组相机参数。扩展存储空间型号(-2)使用 64Mbit(8MB)的 Flash。

| 0x000000 | 0x400000 | 0x7FFFFF |
|---|---|---|
| 程序区 | 用户区 | |

用户可以读写 0x400000-0x7FFFFF 之间的存储区域，共计 4M 字节，其它部分无法读写。

## 2 读写控制

读写的过程需要分两步进行， 使用相机 RAM 作为传输中介传输数据， 如有 RAM 容量有限，可用为 4KB，同时网络传输数据包大小也有限制，每次传输 512 字节。

### 2.1 用户存储数据操作逻辑

1、WRITEMEM 发 8 个 512 字节的包到地址为 0x80000c00 的 RAM 空间，（32bit）其中最高位 1 表示是 RAM 地址而非 GigEVision 地址，一共 4KB 数据量。

2、使用 WRITEMEM 命令 0x05000600 发要保存到的 SPI 地址，注意地址范围只能是 0x400000 0x7FFFFF(8bit),如果保存正常，约 100ms 后收到 ACK，若地址超出范围，立刻收到 ACK（少于 1ms）,若保存失败，约 134ms 收到 ACK。

### 2.2 用户读取数据操作逻辑

1、WRITEMEM 命令发 0x05000604 接 SPI 地址，将 SPI 的 4KB 数据读出到地址为 0x80000c00 的 RAM 空间（32bit），成功则约 2ms 时间收到 ACK,读取 SPI 成功或失败都会收到 ACK，失败则较长时间才收到 ACK。

2、READMEM 命令分 8 次读出 0x80000c00 的 RAM 空间的 4KB 数据。

数据量大于 4KB 时重复以上过程，并实时的修改 SPI 地址。

参考代码：

1 存储数据。

```
//storeToCam
//  将数据写入相机存储空间
//  buffer:   数据缓存器
//  len:      数据长度，写入存储区域需要是4096字节的整数倍， buffer需要留有足够的空间
int storeToCam(void *handle, char *buffer, int len)
{
    int nRet = MV_OK;
    int count = len / 0x200 / 8;

    if(len%0x1000!=0) //填充满4K
    {
        int fillcnt = 0x1000 - len%0x1000;
        memset(buffer+len, 0, fillcnt);
        count += 1;
    }
    for(int i=0; i<count; i++)
    {
        char *p = buffer + i*0x1000;
        for(int j=0; j<8; j++)
        {
            nRet = MV_CC_WriteMemory(handle, p+j*0x200, 0x80000c00+j*0x200/4, 0x200);
            if (MV_OK != nRet)
            {
                printf("WriteMemory fail! nRet [0x%x]\n", nRet);
            }
        }

        int addr = STORE_BASE_ADDR + i*0x1000;
        char addrb[4];
        addrb[0] = (addr&0xff000000) >> 24;
        addrb[1] = (addr&0x00ff0000) >> 16;
        addrb[2] = (addr&0x0000ff00) >> 8;
        addrb[3] = (addr&0x000000ff) >> 0;
        nRet = MV_CC_WriteMemory(handle, addrb, STORE_COMMAND, 4);
        if (MV_OK != nRet)
        {
            printf("WriteMemory fail! nRet [0x%x]\n", nRet);
        }
        else
        {
            printf("Write part %d/%d\n",i+1,count);
        }
    }

    return count*0x1000;
}
```

2 加载数据。

```
//loadFromCam
//  从相机存储空间读取数据
//  buffer:   数据缓存器
//  len:      数据长度，读取存储区域需要是4096字节的整数倍， buffer需要留有足够的空间
int loadFromCam(void *handle, char *buffer, int len)
{
    int nRet = MV_OK;
    int count = len / 0x200 / 8;

    if(len%0x1000!=0) //填充满4K
    {
        count += 1;
    }
    memset(buffer, 0, count*0x1000);

    for(int i=0; i<count; i++)
    {
        int addr = STORE_BASE_ADDR + i*0x1000;
        char addrb[4];
        addrb[0] = (addr&0xff000000) >> 24;
        addrb[1] = (addr&0x00ff0000) >> 16;
        addrb[2] = (addr&0x0000ff00) >> 8;
        addrb[3] = (addr&0x000000ff) >> 0;
        nRet = MV_CC_WriteMemory(handle, addrb, LOAD_COMMAND, 4);
        if (MV_OK != nRet)
        {
            printf("WriteMemory fail! nRet [0x%x]\n", nRet);
        }
        else
        {
            printf("Read part %d/%d\n",i+1,count);
        }

        char *p = buffer + i*0x1000;
        for(int j=0; j<8; j++)
        {
            nRet = MV_CC_ReadMemory(handle, p+j*0x200, 0x80000c00+j*0x200/4, 0x200);
            if (MV_OK != nRet)
            {
                printf("MV_CC_ReadMemory fail! nRet [0x%x]\n", nRet);
            }
        }
    }

    return count*0x1000;
}
```