PCIE2011 RTX 函数库使用说明

1. 变量定义及函数功能

CBissCard 类中主要变量定义如下 ：

m_dwBusNum	总线号（用于搜索设备）
m_dwDeviceNum	设备号
m_dwFunctionNum	功能号（使用时不更改）
m_dwCardNum	卡编号（多个板卡时的板卡虚拟编号）
SlotNumber	PCI 设备槽位号，用于搜索设备
m_clock_freq	板卡采样时钟频率，默认为 2M 时钟
m_protocol	板卡采样协议，默认为 BISS-C
m_clock_bit[5]	板卡各通道采样数据位宽，默认为 26
m_pdwPhyMemory	RTX 连续内存申请的物理地址
m_llMemoryAddr	RTX 地址映射的虚拟地址
vMemAddr[3]	RTX 的 PCI 设备地址映射指针
PciData	PCI 设备配置缓存
m_ppsBissInfo[i][j]	结构体二维数组指针，用于存储采集数据
m_iDataGroupNum	每次采样的数据组数，与 m_ppsBissInfo[i][j]中的 i 对应

注：m_ppsBissInfo[i][j]表示第 j 通道的第 i 组数据，j 范围 0~4 对应 1~5 通道，i 数值越小数据越新，i 范围通过函数 SetDataGroupNum 设置

主要函数定义：

int SearchDeviceBiss(int nCardNumber, PCI_SLOT_NUMBER *pSlotNumber, PPCI_COMMON_CONFIG PciData); 搜索设备，使用时 nCardNumber 自定义，*pSlotNumber 赋值为类内的 SlotNumber 取地址，PciData 赋值为类内的 PciData，函数返回值赋给类内 m_dwBusNum，如错误返回-1，使用示例如下（已实例化类，变量名为 m_BissCard，下同）m_dwCardNum 在构造函数中默认赋值为 0，如使用多个板卡，请在调用该函数前顺序赋值该变量，例：搜索第二张板卡前 m_BissCard_2.m_dwCardNum = 1;

```
// 搜索BissCard
if ((m_BissCard.m_dwBusNum = m_BissCard.SearchDeviceBiss(m_BissCard.m_dwCardNum,&m_BissCard.SlotNumber,m_BissCard.PciData))==-1)
{//not found
    RtPrintf("WARNING: No BISS card 0 was found in this machine.\n");
    ExitProcess(0);
}
else
{
    RtPrintf("BISS card 0 was found in this machine.\n");
}
```

bool InitDeviceBiss(PPCI_COMMON_CONFIG PciData, PCI_SLOT_NUMBER

*pSlotNumber, int nBusNum); 初始化指定板卡,成功返回 true，失败返回 false，使用示例如下

```
//Init device
if(!(m_BissCard.InitDeviceBiss(m_BissCard.PciData, &m_BissCard.SlotNumber,m_BissCard.m_dwBusNum)))
{
    RtPrintf("WARNING: BISS Card 0 Configuration failed!\n");
    ExitProcess(0);
}
else
{
    RtPrintf("BISS Card 0 Configuration OK!\n");
}
```

void SetPcieClock(ULONG clock); 函数 SetPcieClock，设置板卡采样时钟频率，输入参数可用 PCIE_2011_CLOCK_10M 、 PCIE_2011_CLOCK_4M 、PCIE_2011_CLOCK_2M、PCIE_2011_CLOCK_1M、PCIE_2011_CLOCK_500K。

void SetProtocol(ULONG prot); 函数 SetProtocol，设置板卡采样协议，输入参数可用 ENDAT2_1、ENDAT2_2、SSI、BISS_C。

bool SetChannelBit(int Chn, unsigned long bitcount); 函数 SetChannelBit，设置对应通道数据位数（时钟个数），Chn 通道编号（0~4）,bitcount 数据位数，成功返回 true，失败返回 false。

int ConfigDeviceBiss();函数 ConfigDeviceBiss，配置板卡，正常返回 1，返回-1 表示 Rtx 分配连续内存映射失败，返回-2 表示申请内存地址高位异常。

void Snc_CollectBiss();函数 Snc_CollectBiss，向板卡寄存器写命令开始采样

int DMA_TransmissionEnd();函数 DMA_TransmissionEnd，DMA 传输完毕,获取原始数据，原始数据保存在 m_ppsBissInfo[i][j]二维数组中，其中 i 为组数编号（0 为最新），j 为通道编号,正常返回 1，异常返回 0。

void SetDataGroupNum(int iDataGroupNum); 函数 SetDataGroupNum，设置数据组数，默认为 1 组。

void CleanupDeviceBiss();函数 CleanupDeviceBiss，释放板卡资源，程序退出时调用。

int GetCurrentSpeed(double& dCurSpeed, int iChannelNum, bool bIsNeedEWcheck, bool bIsAngle, double dLineRes); 函数 GetCurrentSpeed，获取 iChannelNum(0~4)通道速度（角度返回°/s，直线返回 mm/s），SetDataGroupNum 至少设置 2，bIsNeedEWcheck 设置是否需要 Err 和 War 判断（BISS_C 可用，SSI 一般不可用），bIsAngle 设置编码器是否为角度编码器，角度输入 true 直线输入 false，dLineRes 表示直线光栅分辨率，角度光栅时输入任意值，直线光栅时输入分辨率(um/码值)正常时返回 1，bIsNeedEWcheck 为 true 时若 check 失败返回 0，m_iDataGroupNum 不满足条件返回-1，该通道位数异常返回-2。

int GetCurrentLocation(double& dCurLocation, int iChannelNum, bool bIsNeedEWcheck, bool bIsAngle, double dLineRes); 函数 GetCurrentLocation，iChannelNum 通道当前绝对位置,角度返回°，直线返回 mm，其他参数与 GetCurrentSpeed 相同，数据位数异常返回-2，bIsNeedEWcheck 为 true 时，有通过检测的最新值则正常返回组数索引，即 m_ppsBissInfo[i][j]中的 i，所有组中均未通过返回-1，bIsNeedEWcheck 为 false 时，返回值为 0，dCurLocation 为最新组(0)中数据。

2. 使用流程

方法 1：使用时需将 CBissCard.h 复制并加入 RTX 项目中；将CBissCardRtx.lib 复制到 RTX 项目文件夹中（该方法仅适用 RTX64）并在项目属性→配置属性→链接器→输入→附加依赖项中添加 CBissCardRtx.lib；在RTX 主程序中包含 CBissCard.h 头文件；

方法 2：使用时将 CBissCard.h 和 CBissCard.cpp 加入 RTX 项目中即可。

通过方法 1 或 2 加入 CBissCard 类后，声明 CBissCard 类实体在程序初始化时按如下顺序初始化设备：

- 设置通道数据最大组数，例如：m_BissCard.SetDataGroupNum(1)，建议设置为 1;
- 搜索设备，例：

```
//搜索BissCard
if ((m_BissCard.m_dwBusNum = m_BissCard.SearchDeviceBiss(0, &m_BissCard.SlotNumber, m_BissCard.PciData)) == -1)
{//not found
    RtPrintf("WARNING: No BISS card 0 was found in this machine.\n");
    ExitProcess(0);
}
```

- 初始化设备，例：

```
//Init device
if (!(m_BissCard.InitDeviceBiss(m_BissCard.PciData, &m_BissCard.SlotNumber, m_BissCard.m_dwBusNum)))
{
    RtPrintf("WARNING: BISS Card 0 Configuration failed!\n");
    ExitProcess(0);
}
```

- 配置设备，例：m_BissCard.ConfigDeviceBiss();
- 配置协议、时钟及通道编码器位数，例：

```
m_BissCard.SetProtocol(SSI);
m_BissCard.SetPcieClock(PCIE_2011_CLOCK_500K);
m_BissCard.SetChannelBit(0, 24);
```

上图中 SetProtocol()函数设置采集协议，分别为 BISS_C\SSI\ENDAT；

SetPcieClock()函数设置时钟频率，参数可设置为 PCIE_2011_CLOCK_10M、PCIE_2011_CLOCK_4M 、 PCIE_2011_CLOCK_2M 、 PCIE_2011_CLOCK_1M 、PCIE_2011_CLOCK_500K； SetChannelBit(int i,int j);其中 i 为通道号（0~4），j 为该通道位数；

- 开始采集，例：m_BissCard.Snc_CollectBiss();

初始化完成后，在定时器中获得数据过程如下：

- 传输数据并继续采集，例：

```
m_BissCard.DAM_TransmissionEnd();
m_BissCard.Snc_CollectBiss();
```

- 获得数据并转换，例：

```
for (int i = 0;i<5;i++)
{
    if (    m_pBissCard.m_ppsBissInfo[0][i].b.err
        &&  m_pBissCard.m_ppsBissInfo[0][i].b.war)
    {

        m_psShare->m_dCurrentAngle[i] = 360.00*(m_pBissCard.m_ppsBissInfo[0][i].b.data & 0x03ffffff)/0x03ffffff;//数据换算

        m_psShare->m_dCurrentTime[i] = m_pBissCard.m_ppsBissInfo[0][i].b.div/(1e+2);//时间换算

        m_psShare->m_bChannelStateError[i] = FALSE;
    }
    else
    {
        m_psShare->m_bChannelStateError[i] = TRUE;
    }
}
```

注： m_ppsBissInfo[i][j]联合体中 b 结构体为具体数据定义，包括数据 data:32bit，时间 div:16bit，备用 r1:5bit，校验 crcv:6bit，错误位 err:1bit，警告位 war:1bit，通道号 chn:3bit。其中错误与警告位值 1 为正常，0 为异常。

- 也可通过类内函数获得位置和速度，例：

```
// 数据采集
for (int i = 0; i < 5; i++)//i为通道数
{
    m_BissCard.GetCurrentLocation(m_psShare->m_dCurrentAngle[i], i, true, true, 0);
    m_BissCard.GetCurrentSpeed(m_psShare->m_dCurrentSpeed[i], i, true, true, 0);
}
```

注：以上为获得角编码器位置和速度，且获取速度时 m_iDataGroupNum 需大于等于 2。

- 程序退出时，清理并关闭设备，例：m_BissCard.CleanupDeviceBiss();