VCI_CAN 函数 VB 调用方法简介

函数执行的顺序如下：

1、 打开USB-CAN设备

OpenDevice(DEV_ USBCAN,0,0);

2、初始化CAN控制器

```
Dim pInitInfo As VCI_INIT_CONFIG
pInitInfo.AccCode = 0x80000008;
pInitInfo.AccMask = 0xFFFFFFFF;
pInitInfo.Reserved = 204;
pInitInfo.Filter = 0;
pInitInfo.kCanBaud = 15;
pInitInfo.Timing0 = 0x00;
pInitInfo.Timing1 = 0x14;
pInitInfo.Mode = 0;
VCI_InitCAN(DEV_ USBCAN, 0, 0, pInitInfo);
```

3.启动CAN控制器

```
VCI_StartCAN(DEV_USBCAN, 0, 0);
```

4. CAN总线接收

如果需要比较实时性的接收外部CAN设备发来的数据，可使用定时器中断函数的方式，例如，每隔10ms调用一次接收函数。或者采用多线程方式。

```
PVCI_CAN_OBJ canbuf[60];
int NumValue = VCI_Receive(DEV_USBCAN, 0, 0, canbuf);
```

如果NumValue大于0，表示接受到数据。该值表示返回的CAN帧数，具体数据在canbuf[]数组里面。数组里面保存了收到的CAN帧的格式，ID，数据域字节数，数据内容等

举例：如收到两帧，

则第一帧的ID位于canbuf[0].ID[0…3]，数据位于：canbuf[0].Data[0…7]

则第二帧的ID位于canbuf[1].ID[0…3]，数据位于：canbuf[1].Data[0…7]

5、CAN总线发送

```
PVCI_CAN_OBJ Sendbuf[1];
Sendbuf. RemoteFlag=        //是否是远程帧
Sendbuf.ExternFlag=         //是否是扩展帧
Sendbuf.ID[0]=              //ID域0
Sendbuf.ID[1]=              // ID域1
Sendbuf.ID[2]=              // ID域2
```

```
Sendbuf.ID[3]=           // ID域3
Sendbuf.DataLen＝        //数据域字节长度
Sendbuf.Data[0]=         //数据0
Sendbuf.Data[1]=         //数据1
….
Sendbuf.Data[DataLen-1]=           //数据DataLen-1

VCI_Transmit(DEV_USBCAN, 0, 0, Sendbuf)
```

6、关闭USB-CAN设备

```
CloseDevice(DEV_ USBCAN,0,0);
```

其他还可能用到的函数：

VCI_SetReference(DEV_USBCAN, 0, 0, 0, pData)//该函数用于在工作中进行 CAN 参数修改。如果 InitCAN 已设置 OK，可以不使用这个函数了。和 Initcan 函数功能类似。

详情请仔细阅读说明手册。也可以参考我们提供的 VC 的例程。