附件6：如何更好地使用二次开发函数

说明书

说明书版本：V2.03

更新日期：2017.06.30

1、VCI_OpenDevice 函数

此函数用于连接并打开已插入计算机的USB-CAN适配器。

在通过此函数打开USB-CAN适配器时，所在的计算机进程中将自动生成一个针对USB-CAN进行操作的句柄，并同时在内存中建立相关资源。该函数必须与VCI_CloseDevice成对出现，即：**调用VCI_OpenDevice后，在退出进程、关闭软件、重新打开适配器等情况时，必须调用VCI_CloseDevice函数释放资源，否则可能引起进程崩溃、通信错误等未知错误！**

2、VCI_CloseDevice 函数

此函数用来关闭已打开的USB-CAN适配器，关闭后，适配器将不再进行收发活动，直到下一次打开启动。

该函数应与VCI_OpenDevice函数成对出现，关闭适配器的同时，释放系统资源！

3、VCI_InitCAN 函数

此函数在调用VCI_OpenDevice函数成功之后调用，用来初始化适配器上的一个CAN通道，该函数的形参pInitConfig传递了初始化相关的参数，其中包括滤波参数、波特率、工作模式等。

注意：

1.CAN 总线在正常收发数据的时候，尽量不要通过 USBCAN 适配器修改 CAN 总线参数或关闭 CAN 总线，应等数据收发停止或将 USBCAN 适配器脱离 CAN 总线再进行相应操作。

4、VCI_GetReceiveNum 函数

该函数必须在CAN打开的状态下调用，用来获取在CAN适配器某个通道缓冲区中已经接收到的但未被VCI_Recive函数读取的帧的数量。

在实际编程中，内存一般不会出现紧张，所以一般不用调VCI_GetReceiveNum，以节约CPU资源，而直接调用VCI_Recive，接收用的pReceive结构体数组长度可以设为2000帧pReceive [2000]，每次接收的最大长度Len 也设为2000，Len=2000。如果接收到数据，函数返回相应的帧数，并填充pReceive结构体，如果没有数据，函数返回0。所以不必调用VCI_GetReceiveNum。

5、VCI_StartCAN 函数

该函数用在VCI_InitCAN函数调用成功后，作用是启动一个CAN通道，所启动的CAN通道必须经过VCI_InitCAN函数初始化后才能成功启动，通道启动后，即可通过调用VCI_Receive

和VCI_Transmit函数接收和发送CAN消息。

6、VCI_Receive 函数

该函数必须在CAN通道启动后调用，用来读取CAN通道接收缓存区（设备驱动提供每个通道2000帧CAN消息长度的接收缓冲区，这为应用程序提供了充足的反应处理时间。）中未被读取的CAN消息，它的形参pReceive为接收数据帧数组的首指针；另一个形参len为接收数据帧数组的长度，如果接收缓冲区帧数小于或等于Len时，所有数据均会被读出；如果接收缓冲区帧数大于Len时，Len个数据被读出。所以对应用程序来讲，如果2000帧的数据存储空间不占用太多资源，那么Len可以设为2000及以上，这样能保证每次调用VCI_Receive函数都能把所有数据一次性读回。

在接收线程中，一般通过循环调用VCI_Receive函数来查询接收缓冲区并接收数据。关键参数有：调用频率、接收长度Len。一般情况Len设置成存储空间长度，或者根据实际接收频率及总线消息出现频率设置适当值，前提是这个接收数组要足够大，否则可能出现丢帧现象。

调用频率：调用VCI_Receive函数，实质是读写USB，读写USB有一个最小的耗时，一般为3～5ms，故，VCI_Receive函数调用间隔至少应设置在5ms以上。

调用频率、接收长度Len两个参数直接影响适配器的性能。假设总线速率为9000帧/s，接收长度Len为2000帧，则每秒至少要调用4次，即调用间隔小于250ms。如果接收长度Len设为1000帧，则每秒至少调用8次，即调用间隔小于125ms。否则接收缓冲区会溢出。如此即可匹配调用频率、接收长度Len，得到最优值。

7、VCI_Transmit 函数

该函数必须在CAN通道启动后调用，用来向CAN通道发送缓冲区（USB_CAN适配器提供约每个通道10帧CAN消息长度的发送缓冲区，每次调用VCI_Transmit最多发送约10帧数据。）发送数据，然后通过CAN总线发送出去，它的形参pSend为发送的数据帧数组的首指针；另一个形参len为要发送的数据帧数量，可设置为1-10之间的任意整数。

发送设备的发送速度由当前计算机软硬件性能决定，一般连续发送速度在8000 fps以上(1Mbps)，若发送速度过快将有可能使总线利用率达到100%，使发送方出现发送超时。这样用户可在应用程序中适当添加延时以降低发送速度。

发送过程中每次调用VCI_Transmit函数都有超时限制，发送时超时时间约10ms。发送超时一般由于CAN总线繁忙且当前节点优先级较低时发生，并不是函数调用或通讯错误，用户

可以编程实现重发。因此，在系统设计时注意保证CAN总线占用不应该超过总线容量的60-70%。

调用频率：调用VCI_Transmit函数，实质是读写USB，读写USB有一个最小的耗时，一般为3～5ms，故，VCI_Transmit函数调用间隔至少应设置在5ms以上。

每次发送帧数，建议设为1，即每次发送单帧。每次发送间隔不能太小，否则总线占用率达100%时，发送缓冲区满，由于有堵塞机制，调用VCI_Transmit函数会超时返回，这时候每次调用VCI_Transmit函数都不会立即返回，而是等待10ms直到超时返回，反映到应用程序就是占用资源增加，界面响应迟钝。所以参数设置很重要。

当USB-CAN适配器做为类似于主站的角色时，例如向总线各从节点发送查询指令，再接收各节点返回的数据，每次调用VCI_Transmit函数只能单帧发送，并且要计算好发送间隔，避免总线瞬时负载率过高，导致USBCAN作为发送节点进入关闭状态。如果一次发多帧，那么数据还没有发完，这时总线上面的各从节点已经开始接收到查询指令并返回数据。从节点发送数据与USBCAN作为主站发送数据将同时进行，总线将进入繁忙状态。如果USBCAN一次发送的帧数过多，发送错误计数器将会累加。如果累加过快，达到一定数值时，USBCAN将会进入关闭状态。关闭状态下，未发送的数据会等待发送，不会丢失，USBCAN将接收不到总线上面的数据。所以会导致，查询指令会全部发送给各节点，但是有部分返回的数据将在USBCAN关闭期间丢失，导致USBCAN接收不全。USBCAN关闭后，总线空闲后，会自动回复到正常工作状态，接着发送并接收。这种情况是由CAN总线节点的错误处理机制决定的，只要是CAN节点（USBCAN作为主站也是一个CAN节点，从站也是一个CAN节点）都会有这样的错误处理机制，来避免总线被单个节点占用。所以只能计算好节点数量，查询指令与返回数据的数量，定好负载率，平均分配总线。定好发送间隔。

8、VCI_FindUsbDevice 函数

此函数可以在任意时刻调用，用来查询计算机中所有插入的USB-CAN适配器的信息，并返回前4个适配器的信息，若计算机中插入多于4个适配器，则使用VCI_FindUsbDevice2函数，最大支持50个USB-CAN适配器。