# NDIS 函数简明手册

函数 描述

无连接微端口的上层函数

DriverEntry 由操作系统调用来激活和初始化微端口驱动程序

MiniportAllocateComple 调用它来指示以前调用的 NdisMAllocateMemoryAsync 已经完成

MiniportCheckForHang 检查 NIC 的内部状态

MiniportDisableInterrupt 禁止 NIC 产生中断

MiniportEnableInterrupt 允许 NIC 产生中断

MiniportHalt 重新分配和重新注册 NIC 占用的资源并且终止 NIC 以使它不再有用

MiniportHandleInterrupt 延期用来完成 I/O 中断函数的执行

MiniportInitialize 出始化 NIC

MiniportISR 作为 NIC 中断服务例程以高权值运行

MiniportQueryInformation 查询微端口驱动程序的性能和当前状态

MiniportReconfigure 未使用

MiniportReset 对 NIC 发出一个硬件重置

MiniportReturnPacket 从上层接收一个包，这个包是在这之前通过调用 NdisMIndicateReceivePacket 函数传上去的包

MiniportSend 如果驱动程序没有 MiniportSendPadcets 或 MiniportWanSend 函数，它用来将一个包通过 NIC 传递到网络上

MiniportSendPackets 通过 NIC 传递一组包到网络上

MiniportSetInformation 变换(设置)关于微端口驱动程序或它的 NIC 的信息

MiniportShutdown 当系统暂时关闭时，将 NIC 恢复到初始状态

MiniportSynchronizeISR 同步访问与 MiniportISR 或 MiniportDisableInterrupt 共享的资源。如果有运行在 DIRQL 下的 NIC 驱动程序函数与驱动程序的 MiniportISR 或 MiniportDisableInterrupt 函数共享资源，需要此函数

MiniportTimer 如果微端口的 NIC 不产生中断，用于轮询 NIC 的状态

MiniportTransferData 将由 NIC 接收到的包的内容拷贝到一个给定的包缓存中

MiniportWanSend 如果驱动程序控制着 WAN NIC，通过 NIC 发送一个包到网络上

面向连接微端口的上层函数

DriverEntry 由操作系统调用来激活和初始化微端口驱动程序

MiniportAllocateComplete 调用它来指示以前调用的 NdisMAllocateMemoryAsync 已经完成

MiniportChecKForHang 检查 NIC 的内部状态

MiniportCoActivateVc 激活一个虚连接(VC)

MiniportCoCreateVc 为一个 VC 建立 VC 状态

MiniportCoDeactivateVc 为一个 VC 释放 VC 的状态

MiniportCoDeleteVc 删除一个 VC

MiniportCoRequest 查询微端口驱动程序的能力和当前状态或改变(设置)有关微端口驱动程序或它的 NIC 的信息

MiniportCoSendPackets 通过一个 NIC 发送一组包到网络上

MiniportDisableInterrupt 禁止一个 NIC 产生中断

MiniportEnableInterrupt 允许一个 NIC 产生中断

MiniporeHalt 收回和注销 NIC 占用的资源并且终止 NIC 以使它不再使用

MiniportHandleInterrupt 延期用来完成 I/O 中断函数的执行

MiniportInitialize 初始化 NIC

MiniportISR 作为 NIC 中断服务例程以高权值运行

MiniportReconfigure 未使用

MiniporeReset 对 NIC 发出一个硬件重置

MiniportReturnPacket 从上层接收一个包，这个包是在这以前通过调用 NdisMCoIndicateReceivePacket 传上去的

MiniportShutdown 一个可选择函数，当系统关闭时，将 NIC 恢复到它的初始状态

MiniportSynchronizpISR 同步访问与 MiniportISR 或 MiniportDisableInterrupt 共享的资源，如果有运行在 DIRQL 下的 NIC 驱动程序函数与驱动程序的 MiniportISR 或 MiniportDisableInterrupt 函数共享资源

MiniportTimer 如果微端口 NIC 不产生中断， 轮询一个 NIC 状态

MiniportWanSend 如果驱动程序控制着 WAN NIC ，通过网络接口卡发送一个包到网络上

初始化和注册函数

NdisMInitializeWrapper 为微端口驱动程序初始化 NDIS 库数据结构

NdisMRegisterMiniport 向 NDIS 提供有关微端口驱动程序的信息

NdisMSetAttributes 向 NDIS 库报告微端口驱动程序所支持的 NIC 的类型并且传递指向微端口的环境区域的句柄。NDIS 将在接下来的调用中把这个句柄传递给 MiniportXxx 函数

NdisMSetAtteributesEx 除了如同 NdisMSetAttributes 一样传递同样的信息，这个函数改变了缺省的 NIC 超时动作，并且允许一个微端口指定一些附加的标志。一个驱动程序，例如一个 NDIS 中间层驱动程序或一个非 串行化微端口驱动程序，如果必须指定一个或多个这种标志的话，则必须调用这个函数而不是 NdisMSetAttributes

NdisMGetDeviceProperty 检索出设备对象，这个对象被用来通过一个总线驱动程序来建立一个与 NIC 的通信。例如，USB 和 IEEE 1394 总线驱动程序

NdisMQueryAdapterResources 返回一个 NIC 硬件资源列表

NdisMQueryInformationComplete 指示 NDIS 先前的 MiniporeQueryInformation 操作完成。仅被无连接微端口调用

NdisMSetInformationComplete 报告 NDIS 库，以前的 MiniportSetInofrmation 操作已经完成，仅被无连接微端口调用

NdisOpenConfiguration 提供一个调用者可以用来调用 NdisReadConfiguration，NdisWriteConfiguration，NdisOpenConfigurationByIndex 或 NdisOpenConfigurationByName 的句柄

NdisOpenCanfigurationByIndex 打开一个给定的已打开的注册表中的主键值的一个子键，而这个主键是由一个调用者所提供的句柄指定

NdisOpenConfigarationByName 打开一个给定的已打开的注册表中的主键值的一个指定的子键，这个主键是由一个调用者所提供的句柄指定

NdisReadConfiguration 使用由调用 NdisOpenConfiguration 所获得的参数句柄来读取存储在注册表中的指定键名的键值，键值将传给 NdisReadConfigration

NdisWriteConfiguration 使用由调用 NdisOpenConfiguration 所获得的参数句柄将键值写入注册表，键值传给 NdisWriteConfiguration

NdisCloseConfiguration 关闭一个通过调用 NdisOpenConfiguration 而打开的注册表句柄

NdisMRegisterAdapterShutdownHandler 注册一个 NIC 驱动程序提供的 Miniportshutdown 函数，它在系统将要关闭时被调用

NDIS_INIT_FUNCTION 注明一个驱动程序函数仅在初始化时开始运行

NDIS_PAGEABLE_FUNCTION 注明一个驱动程序函数是可分页代码

硬件设置函数

NdisReadEisaSlotInformation 读取 EISA NIC 插槽信息并且将它拷贝到 NIC 驱动程序所提供的一个缓存中

NdisReadEisaSlotInformationEx 为一个支持可选择设置的 EISA NIC 读取插槽信息和有关的一系列函数的信息，并且将这些设置拷贝到 NIC 驱动程序所提供的缓存中

NdisImmediateReadPciSlotInformation 从一个指定 PCI 设备的设置区间读取一个指定长度的字节

NdisImmediateWritePciSlotInformation 向一个指定 PCI 设备的设置区间写入一个指定长度的字节

NdisMPciAssignResources 返回一个指定 PCI 设备所声明的资源列表

NdisReadPciSlotInformation 从一个指定 PCI 设备的设置区间读取一个指定长度的字节

NdisWritePciSlotInformation 向一个指定 PCI 设备的设置区间写入一个指定长度的字节

NdisReadPcmciaAttributeMemory 为一个 PC 卡的 NIC 从属性内存中读取与总线相关的设置参数

NdisWritePcmciaAttributeMemory 为一个 PC 卡 NIC 向属性内存写入与总线相关的设置参数

NdisReadNetworkAddress 返回软件可设置的网络地址，它是在 NIC 安装在机器中时为 NIC 存储在注册表中的

I/O 端口函数

NdisMRegisterIoPortRange 为使用 NdisRawReadPortXxx 和 NdisRawWriteportXxx 函数建立 I/O 访问端口

NdisMDeregisterIoProtRange 删除早期由 NdisMRegisterIoPortRange 注册的 I/O 访问端口

NdisMMapIospace 为随后的内存映射 I/O 操作映射一定范围的设备内存

NdisZeroMappedMemory 将早期调用 NdisMMapIoSpace 映射的内存块以 0 填充

NdisMoveFromMappedMemory 将在初始化时由 NdisMMapIoSpace 所映射的设备内存中的数据拷贝到一个系统空间的缓存中

NdisMoveToMappedMemory 将数据从一个系统空间缓存中拷贝到在初始化时由 NdisMMapIoSpace 所映射的设备内存中

NdisMUnmapIoSpace 释放调用 NdisMMapIoSpace 所映射的区域

NdisImmediateReadSharedMemory 在驱动程序调用 NdisMMapIoSpace 之前从共享内存地址中读取一块数据到缓存

NdisImmediateWriteSharedMemory 在驱动程序调用 NdisMMapIoSpace 之前向一个共享内存地址中写入缓存中的数据

NdisRawReadPortXxx 从一个 I/O 端口读取唯一的一条数据，具体的函数包括：NdisRawReadPortUchor，NdisRawReadport Ulong，NdisRawReadportUshort

NdisRawReadPortBufferXxx 从一个 I/O 端口一次读取数据到缓存，具体函数包括：NdisRawReadPortBufferUchar，NdisRaw ReadPortBufferUlong，NdisRawReadPortBufferUshort

NdisRawWritePortXxx 向一个 I/O 端口写入一条数据，具体函数包括：NdisRawWritePortUchar，NdisRawWritePortUlong，N disRawWritePortUshourt

NdisRawWritePortBufferXxx 将缓存中的数据写入 I/O 端口，具体函数包括：NdisRawWritePortBufferUch，NdisRawWritePor tBufferUlong，NdisRawWritePortBufferUshort

NdisImmediateReadPortXxx 在驱动程序调用 NdisMRegisterIoPortRange 映射端口之前从一个 I/O 端口读取一条数据。具体函数包括：NdisImmediateReadPortUcha，NdisImmediateReadPortUshort，NdisImmeduateReadPortUlong

NdisImmediateWritePortXxx 在驱动程序调用 NdisMRegisterIoPortRange 映射端口之前向一个 I/O 端口写一条数据。具体函数包括：NdisImmediateWritePortUch，NdisImmediateWritePortUshort，NdisImmediateWritePortUlong

DMA 相关的函数

NdisMAllocateMapRegisters 为使用总线管理器的 DMA 设备分配映射注册表

NdisMAllocateShareMemory 分配系统与一个总线管理器 DMA 和 NIC 共享的内存。它在驱动程序初始化时以 IRQL PASSIV E_LEVEL 调用

NdisMAllocateShareMemoryAsync 调用它提升 IRQL，例如一个微端口的 MiniportHandleInterrupt 函数为总线管理器 DMA NI C 分配共享内存

NdisMCompleteBufferPhysicalMapping 释放早期为一个总线管理器 DMA 操作而调用 NdisMStartBufferPhysicalMapping 所使用的映射注册表。仅在 DMA 初始化时由调用 NdisMAllocateMapRegisters 的驱动程序调用

NdisMCompleteDmaTransfer 指示 NDIS 库一个从属的 DMA 操作已完成

NdisMDeregisterDmaChannel 在微端口驱动程序的 DMA 通道上释放它的声明

NdisFlushBuffer 在发送数据到 NIC 或从 NIC 转移数据之前，调用它来确保在总线管理器 DMA 运行的 cache 和主机物理内存之间一致

NdisGetCacheFillSize 返回微处理器的以字节为单位的 cache 边界。DMA NIC 的驱动程序可以使用由这个函数返回的信息以避免在 DMA 传输时与 cache 断开

NdisMFreeMapRegisters 释放总线管理器 DMA 的映射注册表，它是早期由 NdisMAllocateMapRegisters 分配的

NdisMFreeSharedMemory 释放早期由 NdisMAllocateSharedMemory 或 NdisMAllocateSharedMemoryAsync 分配的内存

NdisMInitializeScatterGatherDma 为使用 DMA 驱动程序保留系统资源，仅被非串行化或面向连接微端口调用

NdisMReadDmaCounter 读取系统 DMA 管理器的计数器的当前值

NdisMRegistgerDmaChannel 为将来从属的 DMA 操作建立一个 DMA 通道控制

NdisMSetupDmaTransfer 为从属的 DMA 传递设置一个主机系统的 DMA 控制器

NdisMStarBufferPhysicalMapping 为一个总线管理器 DMA 操作产生一个物理地址映射，仅在 DMA 初始化时调用 NdisMAllocateMapRegisters 的驱动程序调用

NdisMUpdateSharedMemory 确保在总线管理器 DMA 操作时从共享内存区读取的数据是最新的

NdisQueryMapRegisterCount 返回所有可能的映射注册表数量。在调用 NdisMAllocateMapRegisters 之前驱动程序调用 NdisQueryMapRegisterCount 来选择实际分配多少注册表

中断处理函数

NdisMDeregisterInterrupt 是一个中断处理程序停止接收中断。操作系统断开 NIC 中断相连的中断服务函数的联系

NdisMRegisterInterrupt 连接微端口中断服务函数(MiniportISR)和由它的 NIC 所产生的中断

NdisMSynchronizeWithInterrupt 任何与 MiniportISR 或 MiniportDisableInterrupt 函数共享资源的 NIC 驱动程序函数必须同步访问这些资源以防止争用发生。通过调用 NdisMSynchronizeWithInterrupt 和传递一个 MiniportSynchronizeISR 函数的地址，来使函数与 MiniportISR 和 MiniportDisableInterrupt 同步。MiniportSynchronzeISR 运行在 DIRQL 上，所以它可以安全地访问共享资源

同步函数

NdisMCanceltimer 取消早期由 NdisMSetTimer 设置的一个时钟

NdisMInitializeTimer 初始化一个时钟对象并且将对象与一个 MiniportTimer 函数相连

NdisMSetTimer 设置一个时钟在一个指定间隔之后停止

NdisMSetPeriodicTimer 设置时钟每隔一个指定时间后停止，或直到调用 NdisMCancelTimer 后才停止

NdisMsleep 引起调用者的线程阻塞指定的间隔。驱动程序在初始化或当停止适配器时，调用 NdisMsleep。例如，当等待 NIC 完成初始化时。NdisMsleep 仅在 IRQL PASSIVEL_LEVEL 下被调用

NdisStallExecution 引起调用者的线程停止一个指定间隔，不超过 50 微秒。此时不能使用 NdisMSleep，NdisStallExecution 仅在升高的 IRQL 下被调用

NdisInitializeEvent 产生和初始化一个用来同步驱动程序操作的事件

NdisSetEvent 为指定事件设置信号量

NdisResetEvent 重新设置指定事件为无信号状态

NdisWaitEvent 引起调用者等待到指定事件被指示或指定时间间隔结束时为止

NdisAllocateSpinLock 初始化一个的 NDIS_SPIN_LOCK 类型变量，它被用来同步访问非 ISR 驱动程序函数之间共享的资源

NdisFreeSpinLock 释放一个在调用 NdisAllocateSpinLock 过程中初始化的自旋锁

NdisAcquireSpinLock 获得一个自旋锁来保护在一个 SMP 安全方式下运行的非 ISR 驱动程序函数之间的共享资源的访问。运行在 IRQL<DISPATCH_LEVEL 下的微端口调用这个函数来获得一个自旋锁

NdisReleaseSpinLock 释放一个早期调用 NdisAcquireSpinLock 获得的自旋锁

NdisDprAcquireSpinLock 获得一个在 IRQL DISPATCH_LEVEL 下的自旋锁。它保护在一个 SMP 安全模式下运行的非 ISR 驱动程序函数间的共享资源访问。它比为运行在 IRQL DISPATCH_LEVEL 上的驱动程序函数调用 NdisAcquireSpinLock 要快

NdisDprReleaseSpinLock 释放一个早期调用 NdisDprAcquireSpinLock 获得的自旋锁

NdisInitializeReadWriteLock 初始化一个 NDIS_RW_LOCK 类型变量。NDIS_RW_LOCK 变量用来限制对一个非 ISR 驱动程序线程的共享资源一次进行一个写访问。这个 NDIS_RW_LOCK 允许多个非 ISR 驱动程序线程同时读这些资源。这个读访问在写访问时是不允许的

NdisAcquireReadWriteLock 获得一个调用者用来在多个驱动程序线程的共享资源间进行写或读访问的锁。运行在 IRQL<DISPAICH_LEVEL 下的微端口调用这个函数来获得一个读-写锁。读-写锁适用经常进行读访问但很少进行写访问的资源

NdisReleaseReadWriteLock 释放一个在调用 NdisAcquireReadWriteLock 过程中获得的读-写锁

NdisMSynchronizeWithInterrupt 任何与 MiniportISR 或 MiniportDisableInterrupt 函数共享资源的 NIC 驱动程序函数必须与这两个函数同步访问资源，以防止争用发生。必须与 MiniportISR 和 MiniportDisableInterrupt 同步的函数通过调用 NdisMSynchronizeWithInterrupt，与一个也运行在 DLRQL 的 MiniportSynchronizeISR 函数同步访问共享资源

状态函数

NdisMCoIndicateStatus 向绑定协议指示一个面向连接 NIC 的状态变化或一个在网络接口卡(NIC)上的指定虚连接(VC)的状态变

化

NdisMIndicateStatus 向 NDIS 库指示 NIC 状态已变化，仅被无连接微端口调用

NdisMIndicateStatusComplete 向 NDIS 库指示状态变化已完成，仅被无连接微端口调用

NdisMQueryInformationComplete 指示早期的 MiniportQueryInformation 调用已完成

NdisMSetInformationComplete 指示早期的 MiniportSetInformation 调用已完成

无连接微端口的发送和接收函数

NdisMIndicateReceivePacket 向相关上层指示一个或多个包

NdisMArcIndicateReceiveNdisMEthIndicateReceiveNdisMFddiIndicateReceiveNdisMTrIndicateReceive 指示一个指定介质类型的包正在接收

NdisMArcIndicateReceiveCompleteNdisMEthIndicateReceiveCompleteNdisMFddiIndicateReceiveCompleteNdisMTrIndicateReceiveComplete 指示一个包接收操作已完成

NdisMsendComplete 指示早期包发送操作已完成，当早期 MiniportSend 函数异步操作时使用它

NdisMSendResourcesAvailable 指示 NDIS 库，微端口驱动程序有可用于发送操作的资源。如果微端口完成了一个同步发送，在它的 MiniportSend 函数中调用这个函数，或者当它检测出一个异步发送操作已完成时，在它的 MiniportHandleInterrupt 函数中调用。如果驱动程序没有为一个未完的发送操作调用 NdisMSendComplete 时，NdisMsendResourcesAvailable 才可被调用

NdisMTransferDataComplete 指示 NDIS 库早期的 MiniportTransferDate 请求已完成，当先前调用的 MiniportTransferData 函数返回 NDIS_STATUS_PENDING 时使用这个函数

面向连接微端口的发送和接收函数

NdisMCoIndicateReceivePacket 向上层相关层指示一个或多个包，根据每一个 VC 指示包

NdisMCoSendComplete 指示早期通过调用 MiniportCoSendPackets 所发起的包发送操作已完成

带外(OOB)数据宏

NDIS_OOB_DATA_FROM_PACKET 返回一个指向与一个给定包描述符相连的 OOB 数据块的指针

NDIS_GET_PACKET_MEDIA_SPECIFIC_INFO 从与一个给定包描述符相连的 OOB 数据块中返回 MediaSpecificInformation 指针和 SizeMediaSpecificInfo 值

NDIS_GET_PACKET_STATUS 从一个与给定包描述符相连的 OOB 数据块中返回状态值

NDIS_GET_PACKET_TIME_TO_SEND 从一个与给定包描述符相连的 OOB 数据块中返回 TimeToSend 值

NDIS_SET_PACKET_HEADER_SIZE 为随后的接收指示设置一个与给定包描述符相连的 OOB 数据块中的 HeaderSize 值

NDIS_SET_PACKET_MEDIA_SPECIFIC_LNFO 设置与一个给定包描述符相连的 OOB 数据块中的 MediaSpecificInformation 指针和 SizeMediaSpecificInfo 值

NDIS_SET_PACKET_STATUS 在一个驱动程序调用 NdisM(Co)IndicateReceivePacket 或在一个驱动程序的 Miniport(Co)SendPackets 函数返回控制之前，设置与一个给定包描述符相连的 OOB 数据块中的状态值。

NDIS_SET_PACKET_TIME_RECEIVED 设置与一给定包描述符相连的 OOB 数据块中的 TimeReceived 值

NDIS_SET_PACKET_TIME_SENT 设置与一给定包描述符相连的 OOB 数据块中的 TimeSent 值

包和缓存处理函数

NdisAllocatePacketPool 分配和初始化一个不分页包池块。调用者提供请求可存放包描述符数和以字节为单位的每个固定包的大小

NdisAllocatePacketPoolEx 分配和初始化一个不分页包池的块。并且提供请求包描述符的数量和以字节为单位的每个固定包的大小，调用者提供用来保存溢出状态下的包描述符的数量

NdisAllocatePacket 从由 NdisAllocatePacketPool 返回的包池中分配一固定大小的包描述符

NdisDprAllocatePacket 当调用者运行在 IRQL DISPATCH_LEVEL 下，分配和初始化一个包描述符

NdisAllocateBufferPool 返回一个调用者可用 NdisAllocateBuffer 来分配缓存描述符的句柄

NdisAllocateBuffer 在指定的已分配的不分页的内存块中创建一个映射虚存范围的缓存描述符。需给出由 NdisAllocateBufferPool 返回的句柄

NdisAdjustBufferLength 修改一个给定缓存描述符指定范围的长度

NdisBufferLength 返回一个给定映射缓存以字节为单位的长度

NdisBufferVirtualAddress 如果给定缓存描述符所给出的物理页并没有映射到系统空间的话,返回由一个给定缓存描述符映射的缓存的基本虚地址，并且将这物理页映射到系统空间，

NdisCopyBuffer 为指定的内存块创建缓存描述符

NdisCopyFromPacketToPacket 从一个包向另一个包拷贝指定数量的字节，给定一个指定的源包和一个目的包及每个包的起始位置

NdisCreateLookaheadBufferFromSharedMemory 返回一个与总线管理器 DMA NIC 共享内存块的缓存的虚地址。允许驱动程序以只读预分缓存方式将向上指示的接收数据的一部分映射到相关协议

NdisDestroyLookaheadBufferFromSharedMemory 释放由调用 NdisCreateLookaheadBufferFromSharedMemory 而获得的缓存

NdisQueryPacket 返回一组描述了包和链的大指针的信息

NdisQueryBuffer 返回由一个给定缓存描述符映射的一个缓存的虚地址的基址和长度

NdisQueryBufferOffset 返回在一个给定缓存描述符中映射的一个缓存的虚地址的基址和长度

NdisGetFirstBufferFromPacket 返回链在包上的第一个缓存的描述符及虚地址，以及第一个缓存的长度和所有缓存的总长（以防缓存使分段的）

NdisGetNextBuffer 返回给定缓存描述符链当前的缓存描述符

NdisGetBufferPhysicalArraySize 对一给定缓存描述符，返回不邻接的物理块的缓存数量。总线管理 DMA NIC 在调用 NdisMStarBufferPhysicalMapping 之前调用它来决定分配和填充多少 NDIS_PHYSICAL_ADDRESS 结构

NdisGetPacketFlags 如果有的话，返回在一个给定包中，由协议驱动程序设置的标志

NdisFreePacketPool 释放由调用 NdisAllocatePacketPool 所分配的一个不分页池的块

NdisPacketPoolUsage 返回一个包池中当前已分配的包描述符的数量

NdisFreePacket 释放调用 NdisAllocatePacket 所分配的包

NdisDprFreePacket 当调用者为访问包池提供内部同步，并且调用者运行在 IRQL DISPATCH_LEVEL 时，释放驱动程序分配的包描述符并且将它返回给空闲列表

NdisFreeBufferPool 释放调用 NdisAllocateBufferPool 所获得的句柄

NdisFreeBuffer 释放调用 NdisAllocateBuffer 所获得的缓存描述符

NdisChainBufferAtBack 将一给定缓存描述符链到一个给定包的缓存描述符链表的末尾

NdisUnchainBufferAtFront 从一给定包的缓存描述符链表中删除链表的表头并且返回指向这个缓存描述符的指针

NdisUnchainBufferAtBack 从一给定包的缓存描述符链表中删除表尾，并且返回指向这缓存描述符的指针

NdisReinitializePacket 从一给定包中删除所有相链的缓存并且重新初始化它以备重新使用

NdisRecalculatePacketCounts 重新设置多个给定包的有效个数以使下一个调用 NdisQueryPacket 重新计算个数

NDIS_BUFFER_LINKAGE 由一指向缓存描述符的指针，返回一个被链接的缓存的指针。这个宏允许一个驱动程序不必提供自己的缓存描述符链接情况而对已分配缓存描述符排队

NDIS_BUFFER_TO_SPAN_PAGES 确定给定的缓存取要使用多少物理页

内存函数

NdisAllocateMemory 在指定地址范围地将驻留(不分页)的系统空间内存以物理连续分配方式分配和/或以非 cache 方式分配

NdisAllocateMemorywithTag 如同调用 NdisAllocateMemory，不同在于它允许调用者提供一个标记可用来跟踪驱动程序的内存分配

NdisEqualmemory 将一个内存块中的指定数量的字符与另一个内存块中相同数量的字符相比较

NdisFillMemory 用给定字符填充一个调用者所提供的缓存

NdisFreeMemory 释放早期由 NdisAllocateMemory 分配的内存块

NdisMoveMemory 从一调用者提供的位置将一指定数量的字节拷贝到另一地方

NdisZeroMeomory 用 0 填充内存块

NdisInitializeNPagedLookasideList 初始化一个预先分配的列表，在初始化成功以后，可以从预先分配列表中分配或向列表释放

不分页的固定大小的块

NdisAllocateFromNPagedLookasideList 从一给定的预先分配的列表头中删除第一个项目。如果预先分配列表是空的，那么从不分页池中分配一个项目

NdisDeleteNPagedLookasideList 将一不分页预分配列表从系统中删除

NdisFreeToNPagedLookasideList 返回预分配列表的入口

NdisInitializeListHead 初始化一个顺序的，互锁的，单链表的表头

NdisInterlockedInsertTailList 插入一个项目，通常是一个包，到一个双链表的表尾，这使得可以在一个多处理器安全方式下同步访问列表

NdisInterlockedRemoveHeadList 从表头删除一个项目，这使得可以在一个多处理器安全方式下同步访问列表

NdisInitializeslistHead 初始化一个顺序的，互锁的，单链表的表头

NdisInterlockedPopEntrysList 从一个顺序的，单链表中删除第一个项目

NdisInterlockedPushEntrysList 在一个顺序的，单链表表头插入一个项目

NdisQueryDepthsList 返回一个给定的顺序的，单链表中当前项目数目

系统信息函数

NdisSystemProcessorCount 确定它的调用者运行在一个单处理器上还是多处理机上。一个 NDIS 驱动程序在分配资源之前的初始化中，调用这个函数

NdisGetCurrentProcessorCounts 返回驱动程序当前可使用的处理器的个数，驱动程序用它来确定 CPU 的利用情况

NdisGetCurrentProcessorCpuUsage 以百分比方式返回当前处理器使用率

NdisGetCurrentSystemTime 返回当前系统时间，用来设置时间戳

NdisGetSystemUptime 返回自从系统开始以来所经过的时间值，以微秒为单位

日志函数

NdisMCreateLog 分配和打开一个 NIC 微端口可写入数据的日志文件，这个文件由一个驱动程序提供的 Win32 应用程序显示

NdisMWriteLogData 为供驱动程序提供的 Win32 应用程序使用和显示日志，而将驱动程序提供的信息写入日志文件

NdisMFlushLog 清除日志文件

NdisMCloseLog 释放日志所使用的资源

NdisWriteErroLogEntry 向系统 I/O 出错日志文件写一条目

字符串函数

NdisAnsiStringToUnicodeString 将一给定数量 ANSI 字符串转化为一定数量的 Unicode 字符串

NdisEqualAnsiString 将两个 ANSI 字符串比较并且返回一个表明它们是否相等的值

NdisEqualString 在操作系统缺省的字符集下，比较两个字符串是否相等

NdisEqualUnicodeString 比较两个 Unicode 字符串并且返回一个表明它们是否相等的值

NdisFreeString 释放为缓存字符串分配的存储区

NdisInitAnsiString 初始化一 ANSI 字符串

NdisInitializeString 为一长度字符串分配存储区，并且以系统缺省的字符集初始化它

NdisInitUnicodeString 初始化一 Unicode 字符串

NdisPrintString 在调试窗口显示指定的字符串

NdisUnicodeStringToAnsiString 将一给定的 Unicode 字符串转换成 ANSI 字符串

NdisUpcaseUnicodeString 将一给定 Unicode 字符串副本转换为大写形式，并且返回转换后的字符串

文件函数

NdisOpenFile 返回一个已打开文件的句柄

NdisMapFile 如果文件未被映射，将一个已打开文件映射到一个调用者可访问的缓存中

NdisUnmapFile 释放一个由 NdisMapFile 建立的文件的虚地址映射

NdisCloseFile 释放由 NdisOpenFile 返回的句柄，并且释放文件打开时所分配给文件 contex 的内存

地址函数

NdisGetPhysicalAddressHigh 返回一个给定物理地址的高位部分

NdisGetPhysicalAddressLow 返回一个给定物理地址的低位部分

NdisSetPhysicalAddressHigh 将一个给定物理地址的高位部分设置为一个给定的值

NdisSetPhysicalAddressLow 将一个给定物理地址的低位部分设置为一个给定的值

NDIS_PHYSICAL_ADDRESS_CONST 初始化一个物理地址的静态常量类型

变量函数

NdisRetriveUlong 从源地址中取回一个 ULONG 值，避免对齐错误

NdisStoreUlong 存储一 ULONG 值到一指定地址，避免对齐错误

NdisInterlockedAddUlong 用一个原子操作将一个 Unsigned long 值与一个给定 Unsigned integer 相加，使用一个调用者提供的自旋锁来同步访问 integer 变量

NdisInterlockedDecrement 用一个原子操作减一

NdisInterlockedIncrement 用一个原子操作加一

注册函数

NdisReadRegisterUchar 从一个内存映射设备注册表中读取一个 UCHAR

NdisReadRegisterUlong 从一个内存映射设备注册表中读取一个 ULONG

NdisReadRegiseerUshort 从一个内存映射设备注册表中读取一个 USHORT

NdisWriteRegisterUchar 向一个内存映射设备写入一个 UCHAR

NdisWriteRegisterUlong 向一个内存映射设备写入一个 ULONG

NdisWriteRegisterUshort 向一个内存映射设备写入一个 USHORT

工作项目函数

NdisInitializeWorkItem 当一个系统工作线程获得控制时，用一个调用者提供的环境和回调例程，初始化一个工作队列来作为执行排队

NdisScheduleWorkItem 将一个给定工作项目插入到一个队列。一个系统工作线程删除项目，并且将控制给予驱动程序先前提供给 NdisInitializeWorkItem 的回调函数

NDIS 提供的媒体相关宏

ETH_COPY_NETWORK_ADDRESS 将一个给定 Ethernet 地址拷贝到一个给定位置

FDDI_IS_BROADCAST 将一个调用者提供的变量设置为一个布尔值，它表示一个给定的 FDDI 地址是否是一个广播地址

FDDI_IS_MULTICAST 将一个调用者提供的变量设置为一个布尔值，它表示一个给定的 FDDI 地址是否是多点传输地址

FDDI_IS_SMT 设置一个调用者提供的变量为一布尔值，它表示是否给定的 FDDI 帧是 SMT

TR_COMPARE_NETWORK_ADDRESS 将一个调用者提供的变量设置为一个布尔值，表明一个给定的 Token Ring 地址是大于、小于或等于另一个给定 Token Ring 地址的值

TR_COPY_NETWORK_ADDRESSES 将一个给定 Token Ring 地址拷贝到一给定位置

IR_IS_BROADCAST 将一个调用者提供的变量设置为一个布尔值，它表示一个给定的 Token Ring 地址是否是广播地址

TR_IS_FUNCTIONAL 将一个调用者提供的变量设置为一个布尔值，它表示一个给定 Token Ring 地址是否是函数地址

TR_IS_GROUP 将一个调用者提供的变量设置为一个布尔值，它表示一个给定 Token Ring 地址是否是一组地址

TR_IS_NOT_DIRECTED 将一个调用者提供的变量设置为一个布尔值，它表示一给定 Token Ring 地址是否既不是一个函数地址也不是一组地址

TR_IS_SOURCE_ROUTING 将一个调用者提供的变量设置为一个布尔值，它表示一给定 Token Ring 地址是否是一源路由地址