NDAS Communication API Ver 0.3

Table of Contents

1 Overview 1

1.1 NDAS API Architecture 2

1.2 NDAS Device 3

1.2.1 NDAS Device Version 4

1.2.2 NDAS Unit Device 6

1.3 LPX Protocol 7

1.4 NDAS Communication API 8

1.4.1 About NDASComm 9

1.4.2 Using NDASComm 10

2 Symbol Reference 13

2.1 Functions 14

2.1.1 NdasCommBlockDeviceRead 15

2.1.2 NdasCommBlockDeviceVerify 16

2.1.3 NdasCommBlockDeviceWrite 17

2.1.4 NdasCommBlockDeviceWriteSafeBuffer 18

2.1.5 NdasCommConnect 19

2.1.6 NdasCommDisconnect 21

2.1.7 NdasCommGetAPIVersion 22

2.1.8 NdasCommGetDeviceID 23

2.1.9 NdasCommGetDeviceInfo 24

2.1.10 NdasCommGetHostAddress 25

2.1.11 NdasCommGetTransmitTimeout 26

2.1.12 NdasCommGetUnitDeviceInfo 27

2.1.13 NdasCommGetUnitDeviceStat 28

2.1.14 NdasCommIdeCommand 29

2.1.15 NdasCommInitialize 31

2.1.16 NdasCommSetFeatures 32

2.1.17 NdasCommSetTransmitTimeout 33

2.1.18 NdasCommUninitialize 34

2.1.19 NdasCommVendorCommand 35

2.2 Structs, Records, Enums 37

2.2.1 NDASCOMM_BIN_PARAM_TARGET_DATA 38

2.2.2 NDASCOMM_VCMD_PARAM 39

2.2.3 NDASCOMM_BIN_PARAM_TARGET_LIST 43

2.2.4 NDASCOMM_BIN_PARAM_TARGET_LIST_ELEMENT 44

2.2.5 NDASCOMM_CONNECTION_INFO 45

2.2.6 NDASCOMM_HANDLE_INFO_TYPE 47

2.2.7 NDASCOMM_IDE_REGISTER 49

2.2.8 NDASCOMM_UNIT_DEVICE_INFO 52

2.2.9 NDASCOMM_UNIT_DEVICE_STAT 53

2.3 Types 54

2.3.1 NDASCOMM_VCMD_COMMAND 55

2.3.2 NDASCOMM_HANDLE_INFO_TYPE 57

3 Index 59

NDAS Communication API Ver 0.3

1 Overview

Purpose

NDAS Communication API (NDASCOMM) enables programmers to interact with a NDAS device directly With NDASCOMM, programmers can operate not only basic operations such as reading or writing, but also advanced operations including most parts of the ATA commands such as identifying or setting features.

Developer Audience

NDAS API's are designed for use by C/C++ programmers, Familiarity with networking, NDAS devices and ATA commands (to use make use of advanced features of NDASCOMM) is required.

Run-time Requirements

NDAS API requires Windows 2000, Windows XP or Windows Server 2003 and a system with LPX protocol and Windows Socket 2.

NDAS Communication API

NDAS Communication API	Description
About NDASComm (see page 9)	**NDAS Communication API** NDAS Communication API(NDASComm) provides functions to communicate with NDAS Device(s) with LPX protocol over Ethernet. NDASComm does not only provide basic functions that application can send ATA command to the NDAS Device but also utility functions to read, write, verify or identify the NDAS Unit Device. This library is provided to support manufacturing processes and for diagnostic purpose. Do not redistribute this library.
Using NDASComm (see page 10)	The following is a step-by-step guide to getting started with NDASCOMM library. It is designed to provide an understanding of basic NDASComm functions and data structures, and how they work together. **Creating a Basic NDASCOMM Application** To create a basic NDASCOMM application

1.1 NDAS API Architecture

Figure 1 describes the architecture of NDAS API.

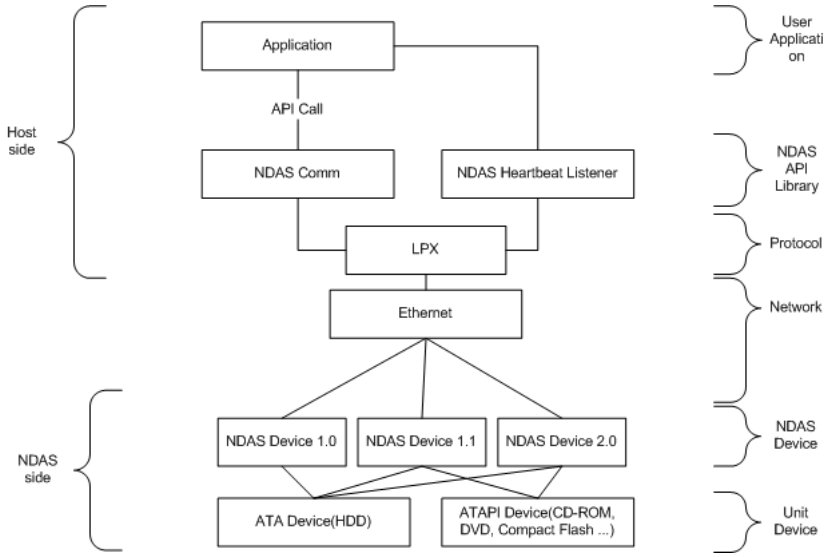

Figure 1. Architecture of NDAS API.

NDAS Communication Library translates API calls into NDAS device commands and sends them to a NDAS device with LPX protocol over Ethernet.

NDAS Heartbeat Listener Library listens a designated heartbeat packet from NDAS devices in the network and calls the user-supplied callback function with a NDAS device information.

1.2 NDAS Device

NDAS (Network Direct Attached Storage) is XIMETA's patented technology which enables all digital storage devices (HDD, ODD, Memory, Tape Drives) connect into standard Ethernet network and all the users or systems in the network directly control, use and share those devices.

1.2.1 NDAS Device Version

Up to now, three versions of the NDAS devices are released.

NDAS device 1.0

The NDAS device 1.0 provides following ATA commands:

- READ(0x20)
- WRITE(0x30)
- VERIFY(0x40)
- IDENTIFY(0xFC)
- SET FEATURE(0xEF)

Internally, NDAS device 1.0 does not support direct ATA commands, but it translates pre-defined command set into actual ATA comands. It is not recommanded to use NdasCommIdeCommand (see page 29) function with NDAS device 1.0.

Characteristics of NDAS device 1.0 are as follows:

- supports ethernet connection up to 100 Mbps.
- supports Multiword DMA level 0, 1 and 2.
- supports ATA devices.
- does not support ATAPI devices.

NDAS Device Ver 1.1

NDAS Device 1.1 supports virtually all ATA commands including packet commands (but NDASCOMM dose not supports packet command at this time). The NDAS Device 1.1 receives an ATA command and send it directly to the NDAS unit device.

Characteristics of NDAS device 1.1 are as follows:

- supports ethernet connection up to 100 Mbps.
- supports Multiword DMA level 0, 1 and 2.
- supports ATA devices.
- supports ATAPI devices.

NDAS Device Ver 2.0

NDAS Device 2.0 has a same ATA command set as NDAS device 1.1.

See the table for differences.

Characteristics of NDAS device 2.0 are as follows:

- supports ethernet connection up to 1 Gbps.
- supports Multiword DMA level 0, 1 and 2.
- supports Ultra DMA level 0 to 6.
- supports ATA devices.
- supports ATAPI devices.

Differences by the version of the NDAS Device.

	1.0	**1.1**	**2.0**
Max Speed	100 MBps	100 MBps	1 GBps
Unit Device	ATA Device	ATA/ATAPI Deivce	ATA/ATAPI Device
DMA level	Multiword DMA 0, 1 and 2	Multiword DMA 0, 1 and 2	Multiword DMA 0, 1 and 2 Ultra DMA 0 to 6

Command	READ, WRITE, VERIFY, IDENTIFY, SET FEATURE	All ATA Commands	All ATA Commands

1.2.2 NDAS Unit Device

NDAS unit device (or unit device) is an ATA/ATAPI device which is attached to a NDAS device so that a host can access to the device via NDAS device.

The number of maximum unit devices that a NDAS device supports can be obtained by NdasCommGetDeviceInfo (see page 24).

1.3 LPX Protocol

Short for Lean Packet eXchange Protocol. It is defined by XIMETA, Inc. and used to communicate with NDAS device over the Ethernet. It is a light-weight and efficient protocol to provide high performance data transfer between a NDAS host and a NDAS device.

1.4 NDAS Communication API

1.4.1 About NDASComm

NDAS Communication API

NDAS Communication API(NDASComm) provides functions to communicate with NDAS Device(s) with LPX protocol over Ethernet.

NDASComm does not only provide basic functions that application can send ATA command to the NDAS Device but also utility functions to read, write, verify or identify the NDAS Unit Device.

This library is provided to support manufacturing processes and for diagnostic purpose. Do not redistribute this library.

1.4.2 Using NDASComm

The following is a step-by-step guide to getting started with NDASCOMM library. It is designed to provide an understanding of basic NDASComm functions and data structures, and how they work together.

Creating a Basic NDASCOMM Application

To create a basic NDASCOMM application

1. Create a new empty project.
2. Set the build environment to include paths for the headers and libraries.
3. Set link options to link against ndascomm.lib.
4. Include "ndascomm.h" to use NDASCOMM API.
5. Ensure to place ndascomm.dll in an NDASCOMM application's search path.

```
#include <ndascomm.h>

int main()
{
  printf("Starting NDAS application\n");
  return 0;
}
```

Initializing NDASComm

All NDASComm applications should initialize NDASCOMM library first before calling other API functions to ensure that NDASComm is supported on the system.

To Initialize NDASComm library

- Call NdasCommInitialize (see page 31) and check errors

```
BOOL bResult = NdasCommInitialize();
if (!bResult)
{
  printf("Error %d at NdasCommInitialize()\n", GetLastError());
  return FALSE;
}
```

The NdasCommInitialize (see page 31) function is called to initiate use of ndascomm.lib

```
DWORD dwVersion = NdasCommGetAPIVersion();
WORD wMajorVersion = LOWORD(dwVersion);
WORD wMinororVersion = HIWORD(dwVersion);
```

The NdasCommGetAPIVersion (see page 22) function is called to retrieve the version of ndascomm.lib

Creating connection to an NDAS Device

After initialization, a NDASCOMM_CONNECTION_INFO struct must be initialized.

```
NDASCOMM_CONNECTION_INFO m_ci;
ZeroMemory(&m_ci, sizeof(NDASCOMM_CONNECTION_INFO));

/* address using ansi code ID string */
m_ci.address_type = NDASCOMM_CONNECTION_INFO_TYPE_ID_A;

/* operates active jobs */
m_ci.login_type = NDASCOMM_LOGIN_TYPE_NORMAL;

/* use first unit device */
m_ci.UnitNo = 0;

/* login with write privilage */
m_ci.bWriteAccess = TRUE;

/* use default normal user password */
m_ci.ui64OEMCode = NULL;

/* connect with LPX protcol */
m_ci.protocol = IPPROTO_LPXTCP;

/* set 20 chars of ansi code ID */
CopyMemory(m_ci.sDeviceStringId, "ABCDE12345EFGHI67890", 20);

/* set 5 chars of ansi write key */
CopyMemory(m_ci.sDeviceStringKey, "12345", 5);
```

After NDASCOMM_CONNECTION_INFO struct initialized, Connect to the NDAS Device using NdasCommConnect (see page 19).

```
HNDAS hNDASDevice = NdasCommConnect(&ConnectionInfo);

if(!hNDASDevice)
{
  printf("Error %d at NdasCommConnect()\n", GetLastError());
  return FALSE;
}
```

Retrieving informations from an NDAS Device and/or Unit Device

See examples of NdasCommGetDeviceID (see page 23)(), NdasCommGetDeviceInfo (see page 24)(), NdasCommGetUnitDeviceInfo (see page 27)()

Operates basic commands

See examples of NdasCommBlockDeviceRead (see page 15)(), NdasCommBlockDeviceWrite (see page 17)(), NdasCommBlockDeviceVerify (see page 16)(), NdasCommSetFeatures (see page 32)(), NdasCommBlockDeviceWriteSafeBuffer (see page 18)()

Operates advanced commands

See examples of NdasCommIdeCommand (see page 29)(), NdasCommVendorCommand (see page 35)()

Disconnecting connection from an NDAS Device

The NdasCommDisconnect (see page 21) function is called after the use of the NDAS Device.

```
bResult = NdasCommDisconnect(hNDASDevice);
if(!bResult)
{
  printf("Error %d at NdasCommDisconnect()\n", GetLastError());
  return FALSE;
}
```

2 Symbol Reference

Functions

Function	Description
NdasCommBlockDeviceRead (see page 15)	The NdasCommBlockDeviceRead function reads data from the unit device.
NdasCommBlockDeviceVerify (see page 16)	The NdasCommBlockDeviceVerify function verifies the unit device.
NdasCommBlockDeviceWrite (see page 17)	The NdasCommBlockDeviceWrite function writes data to a unit device.
NdasCommBlockDeviceWriteSafeBuffer (see page 18)	The NdasCommBlockDeviceWriteSafeBuffer function writes data to a unit device without corrupting data buffer.
NdasCommConnect (see page 19)	The NdasCommConnect function connects to a NDAS Device using given information.
NdasCommDisconnect (see page 21)	The NdasCommDisconnect function closes connection to NDAS Device.
NdasCommGetAPIVersion (see page 22)	The NdasCommGetAPIVersion function returns the current version information of the loaded library
NdasCommGetDeviceID (see page 23)	The NdasCommGetDeviceID function retrieve device ID and unit number from the NDAS Device.
NdasCommGetDeviceInfo (see page 24)	The NdasCommGetDeviceInfo function retrieves static NDAS Device informations from the NDAS Device.
NdasCommGetHostAddress (see page 25)	The NdasCommGetHostAddress function retrieves the host address which is connected to the NDAS Device.
NdasCommGetTransmitTimeout (see page 26)	The NdasCommGetTransmitTimeout retrieves the transmit timeout value.
NdasCommGetUnitDeviceInfo (see page 27)	The NdasCommGetUnitDeviceInfo retrieves static Unit Device information from the NDAS Device.
NdasCommGetUnitDeviceStat (see page 28)	The NdasCommGetUnitDeviceStat retrieves dynamic Unit Device information from the NDAS Device.
NdasCommIdeCommand (see page 29)	The NdasCommIdeCommand sends ide command and receives the result.
NdasCommInitialize (see page 31)	The NdasCommInitialize function initiates use of NdasComm.DLL or NdasComm.LIB by a process.
NdasCommSetFeatures (see page 32)	The NdasCommSetFeatures set features of the unit device.
NdasCommSetTransmitTimeout (see page 33)	The NdasCommSetTransmitTimeout sets the transmit timeout value.
NdasCommUninitialize (see page 34)	The NdasCommUninitialize function makes end use of NdasComm.DLL or NdasComm.LIB by a process.
NdasCommVendorCommand (see page 35)	The NdasCommVendorCommand sends vendor specific command and receives the results.

Types

Type	Description
NDASCOMM_VCMD_COMMAND (see page 55)	Specifies the vendor command type parameter for NdasCommVendorCommand (see page 35)
NDASCOMM_HANDLE_INFO_TYPE (see page 57)	Specifies the information type to retrieve from the handle to the NDAS Device.

Structs, Records, Enums

Struct, Record, Enum	Description
_NDASCOMM_BIN_PARAM_TARGET_DATA (see page 38)	Binary type parameter for text request command. Not used. Use NdasCommGetUnitDeviceStat (see page 28) function to retrieve the information instead.
_NDASCOMM_VCMD_PARAM (see page 39)	Specifies the attributes of connection information. This structure is used to create connection to a NDAS Device.
_NDASCOMM_BIN_PARAM_TARGET_LIST (see page 43)	Binary type parameter for text request command. Not used. Use NdasCommGetUnitDeviceStat (see page 28) function to retrieve the information instead.
_NDASCOMM_BIN_PARAM_TARGET_LIST_ELEMENT (see page 44)	Binary type parameter for text request command. Not used. Use NdasCommGetUnitDeviceStat (see page 28) function to retrieve the information instead.
_NDASCOMM_CONNECTION_INFO (see page 45)	Specifies the attributes of connection information. This structure is used to create connection to a NDAS Device.
_NDASCOMM_HANDLE_INFO_TYPE (see page 47)	Specifies the information type to retrieve from the handle to the NDAS Device.
_NDASCOMM_IDE_REGISTER (see page 49)	Specifies and receives IDE command register information to the unit device attached the NDAS Device. Fill registers according to command member. reg, device and command members are bi-directional. So, those will be filled by return register informations.
_NDASCOMM_UNIT_DEVICE_INFO (see page 52)	Receives the static attributes of unit device attached to the NDAS Device. All attributes are obtained by WIN_IDENTIFY or WIN_PIDENTIFY IDE command. Using NdasCommIdeCommand (see page 29), same information can be obtained also.
_NDASCOMM_UNIT_DEVICE_STAT (see page 53)	Receives the dynamic attributes of unit device attached to the NDAS Device.

2.1 Functions

2.1.1 NdasCommBlockDeviceRead

The NdasCommBlockDeviceRead function reads data from the unit device.

```
NDASCOMM_API BOOL NDASAPICALL NdasCommBlockDeviceRead(
    IN HNDAS hNDASDevice,
    IN INT64 i64Location,
    IN UINT64 ui64SectorCount,
    OUT PBYTE pData
);
```

File

ndascomm.h

Parameters

Parameters	Description
`IN HNDAS hNDASDevice`	Handle to the NDAS Device.
`IN INT64 i64Location`	Sector position of the NDAS Device to read data from. If value is negative, the function counts back from the last sector. (ex: -1 indicates the last sector)
`IN UINT64 ui64SectorCount`	Number of data size in sectors to read.
`OUT PBYTE pData`	A pointer to the read data buffer.

Returns

If the function succeeds, the return value is non-zero. If the function fails, the return value is zero. To get extended error information, call GetLastError.

Description

The NdasCommBlockDeviceRead function reads data from the unit device. The function sends one of WIN_READ, WIN_READDMA, WIN_READ_EXT and WIN_READDMA_EXT by unit device's DMA, Ultra DMA and LBA48 capability. The function should not be called to read data from a non-block device or non LBA supporting block device.

Example

```
// Reads 4 sectors of data from sector 10 of unit device. Suppose hNDASDevice is valid
handle to a NDAS Device.
BOOL bResult;
CHAR data_buffer[4 * 512];

bResult = NdasCommBlockDeviceRead(hNDASDevice, 10, 4, data_buffer);
```

2.1.2 NdasCommBlockDeviceVerify

The NdasCommBlockDeviceVerify function verifies the unit device.

```
NDASCOMM_API BOOL NDASAPICALL NdasCommBlockDeviceVerify(
    IN HNDAS hNDASDevice,
    IN INT64 i64Location,
    IN UINT64 ui64SectorCount
);
```

File

ndascomm.h

Parameters

Parameters	Description
`IN HNDAS hNDASDevice`	Handle to the NDAS Device.
`IN INT64 i64Location`	Sector position of the Unit Device to verify. If value is negative, the function counts back from the last sector. (ex: -1 indicates the last sector)
`IN UINT64 ui64SectorCount`	Number of data size in sectors to verify.

Returns

If the function succeeds, the return value is non-zero. If the function fails, the return value is zero. To get extended error information, call GetLastError.

Description

The NdasCommBlockDeviceVerify function verifies the unit device. The function sends one of WIN_VERIFY, WIN_VERIFY_EXT by device's LBA48 capability. The function should not be called to verify a non-block device or non LBA supporting block device.

2.1.3 NdasCommBlockDeviceWrite

The NdasCommBlockDeviceWrite function writes data to a unit device.

```
NDASCOMM_API BOOL NDASAPICALL NdasCommBlockDeviceWrite(
    IN HNDAS hNDASDevice,
    IN INT64 i64Location,
    IN UINT64 ui64SectorCount,
    IN OUT PBYTE pMutableData
);
```

File

ndascomm.h

Parameters

Parameters	Description
`IN HNDAS hNDASDevice`	Handle to the NDAS Device.
`IN INT64 i64Location`	Sector position of the Unit Device to write data to. If value is negative, the function counts back from the last sector. (ex: -1 indicates the last sector)
`IN UINT64 ui64SectorCount`	Number of data size in sectors to write.
`IN OUT PBYTE pMutableData`	A pointer to the write data buffer. Contents in buffer will be filled with invalid data after function returns. Use NdasCommBlockDeviceWriteSafeBuffer (see page 18) function to protect buffer.

Returns

If the function succeeds, the return value is non-zero. If the function fails, the return value is zero. To get extended error information, call GetLastError.

Description

The NdasCommBlockDeviceWrite function writes data to a unit device. The function sends one of WIN_WRITE, WIN_WRITEDMA, WIN_WRITE_EXT and WIN_WRITEDMA_EXT by unit device's DMA, Ultra DMA and LBA48 capability. The function should not be called to write data to a non-block device or non LBA supporting block device.

See Also

NdasCommBlockDeviceWriteSafeBuffer (see page 18)

Example

```
// Writes 2 sectors of data to last 2 sectors of unit device. Suppose hNDASDevice is valid
handle to a NDAS Device.
BOOL bResult;
CHAR data_buffer[2 * 512];

bResult = NdasCommBlockDeviceWrite(hNDASDevice, -2, 2, data_buffer);
```

2.1.4 NdasCommBlockDeviceWriteSafeBuffer

The NdasCommBlockDeviceWriteSafeBuffer function writes data to a unit device without corrupting data buffer.

```
NDASCOMM_API BOOL NDASAPICALL NdasCommBlockDeviceWriteSafeBuffer(
    IN HNDAS hNDASDevice,
    IN INT64 i64Location,
    IN UINT64 ui64SectorCount,
    IN CONST BYTE* pData
);
```

File

ndascomm.h

Parameters

Parameters	Description
`IN HNDAS hNDASDevice`	Handle to the NDAS Device.
`IN INT64 i64Location`	Sector position of the Unit Device to write data to. If value is negative, the function counts back from the last sector. (ex: -1 indicates the last sector)
`IN UINT64 ui64SectorCount`	Number of data size in sectors to write.
`IN CONST BYTE* pData`	A pointer to the write data buffer.

Returns

If the function succeeds, the return value is non-zero. If the function fails, the return value is zero. To get extended error information, call GetLastError.

Description

The NdasCommBlockDeviceWriteSafeBuffer does act same way as NdasCommBlockDeviceWrite (see page 17) function does except one thing. While the NdasCommBlockDeviceWrite (see page 17) function corrupts data buffer after the function returns, the NdasCommBlockDeviceWriteSafeBuffer function does not.

See Also

NdasCommBlockDeviceWrite (see page 17)

2.1.5 NdasCommConnect

The NdasCommConnect function connects to a NDAS Device using given information.

```
NDASCOMM_API HNDAS NDASAPICALL NdasCommConnect(
    IN CONST NDASCOMM_CONNECTION_INFO* pConnectionInfo,
    IN CONST DWORD dwTimeout,
    IN CONST VOID* hint
);
```

File

ndascomm.h

Parameters

Parameters	Description
`IN CONST NDASCOMM_CONNECTION_INFO* pConnectionInfo`	A connection information to a NDAS Device.
`IN CONST DWORD dwTimeout`	Indicator specifying the time-out interval, in milliseconds. If dwTimeout is 0, the function's time-out interval never expires.
`IN CONST VOID* hint`	If hint is not NULL, the function tries the host addresses in hint . If hint is NULL, the function tries all the host addresses until the connection established. If protocol is NDASCOMM_TRANSPORT_LPX, hint should be LPSOCKET_ADDRESS_LIST type.

Returns

If the function succeeds, the return value is an open handle to the NDAS Device.

If the function fails, the return value is zero. To get extended error information, call GetLastError.

Description

The NdasCommConnect function connects to a NDAS Device using given connection information. The NdasCommConnect function connects to the NDAS Device and process login with given login information. When login process succeeds and if the NDAS Device is a block device(non-packet device), the NdasCommconnect function process handshake to the connect and retrieve DMA, LBA, LBA48 information to the handle which are used to read/write/verify the NDAS Device. Re-handshaking is available using NdasCommIdeCommand (see page 29) but handshake information is not stored to the handle.

See Also

NdasCommBlockDeviceRead (see page 15), NdasCommBlockDeviceVerify (see page 16), NdasCommBlockDeviceWrite (see page 17), NdasCommBlockDeviceWriteSafeBuffer (see page 18), NdasCommDisconnect (see page 21), _NDASCOMM_CONNECTION_INFO (see page 45)

Example

```
// Connects to and disconnects from a NDAS Device

BOOL bResult;
NDASCOMM_CONNECTION_INFO ConnectionInfo;
HNDAS hNDASDevice;

ZeroMemory(&ConnectionInfo, sizeof(NDASCOMM_CONNECTION_INFO));
ConnectionInfo.address_type = NDASCOMM_CONNECTION_INFO_TYPE_ID_A;
ConnectionInfo.login_type = NDASCOMM_LOGIN_TYPE_NORMAL;
ConnectionInfo.UnitNo = 0;
ConnectionInfo.bWriteAccess = TRUE;
ConnectionInfo.ui64OEMCode = NULL;
ConnectionInfo.protocol = NDASCOMM_TRANSPORT_LPX;
memcpy(ConnectionInfo.sDeviceStringId, "ABCDE12345EFGHI67890", 20);
memcpy(ConnectionInfo.sDeviceStringKey, "12345", 5);

hNDASDevice = NdasCommConnect(&ConnectionInfo, 0, NULL);

if(!hNDASDevice)
  return FALSE; // failed to create connection
```

```
bResult = NdasCommDisconnect(hNDASDevice);

if(!bResult)
  return FALSE;

return TRUE;
```

2.1.6 NdasCommDisconnect

The NdasCommDisconnect function closes connection to NDAS Device.

```
NDASCOMM_API BOOL NDASAPICALL NdasCommDisconnect(
    IN HNDAS hNDASDevice
);
```

File

ndascomm.h

Parameters

Parameters	Description
`IN HNDAS hNDASDevice`	Handle to the NDAS Device.

Returns

If the function succeeds, the return value is non-zero. If the function fails, the return value is zero. To get extended error information, call GetLastError.

Description

The NdasCommDisconnect function closes connection to NDAS Device. The NdasCommDisconnect function try to logout from the NDAS Device. And whether logout process succeed or not, the function disconnects connection.

See Also

NdasCommConnect (see page 19)

2.1.7 NdasCommGetAPIVersion

The NdasCommGetAPIVersion function returns the current version information of the loaded library

```
NDASCOMM_API DWORD NDASAPICALL NdasCommGetAPIVersion();
```

File

ndascomm.h

Returns

Lower word contains the major version number and higher word the minor version number.

Description

The NdasCommGetAPIVersion function returns the current version information of the loaded library

2.1.8 NdasCommGetDeviceID

The NdasCommGetDeviceID function retrieve device ID and unit number from the NDAS Device.

```
NDASCOMM_API BOOL NDASAPICALL NdasCommGetDeviceID(
    IN HNDAS hNDASDevice,
    OUT PBYTE pDeviceId,
    OUT LPDWORD pUnitNo
);
```

File

ndascomm.h

Parameters

Parameters	Description
IN HNDAS hNDASDevice	Handle to the NDAS Device.
OUT PBYTE pDeviceId	Pointer to a BYTE array to retrieve device ID.
OUT LPDWORD pUnitNo	Pointer to a BYTE variable to retrive unit number.

Returns

If the function succeeds, the return value is non-zero. If the function fails, the return value is zero. To get extended error information, call GetLastError.

Description

The NdasCommGetDeviceID function retrieve device ID and unit number from the NDAS Device.

Example

```
// retrieve NDAS Device ID & unit number

BOOL bResult;
BYTE DeviceID[6];
BYTE UnitNo;

bResult = NdasCommGetDeviceID(hNDASDevice, DeviceID, &UnitNo);
```

2.1.9 NdasCommGetDeviceInfo

The NdasCommGetDeviceInfo function retrieves static NDAS Device informations from the NDAS Device.

```
NDASCOMM_API BOOL NDASAPICALL NdasCommGetDeviceInfo(
    IN HNDAS hNDASDevice,
    IN NDASCOMM_HANDLE_INFO_TYPE info_type,
    OUT PBYTE data,
    IN UINT32 data_len
);
```

File

ndascomm.h

Parameters

Parameters	Description
IN HNDAS hNDASDevice	Handle to the NDAS Device.
IN NDASCOMM_HANDLE_INFO_TYPE info_type	Handle information type to retrieve.
OUT PBYTE data	Pointer to a data buffer to store the information.
IN UINT32 data_len	A size of the data buffer.

Returns

If the function succeeds, the return value is non-zero. If the function fails, the return value is zero. To get extended error information, call GetLastError.

Description

The NdasCommGetDeviceInfo function retrieves static NDAS Device handle informations from the NDAS Device. A static information is not changed as long as the NDAS Device is connected. See NDASCOMM_HANDLE_INFO_TYPE (see page 57) for detailed information.

See Also

_NDASCOMM_HANDLE_INFO_TYPE (see page 47)

Example

```
// retrieve maximum transfer blocks that the NDAS Device supports.

BOOL bResult;
UINT32 MaxBlocks;

bResult = NdasCommGetDeviceInfo(hNDASDevice, ndascomm_handle_info_hw_max_blocks,
&MaxBlocks, sizeof(MaxBlocks));
```

2.1.10 NdasCommGetHostAddress

The NdasCommGetHostAddress function retrieves the host address which is connected to the NDAS Device.

```
NDASCOMM_API BOOL NDASAPICALL NdasCommGetHostAddress(
    IN HNDAS hNDASDevice,
    OUT PBYTE Buffer,
    IN OUT LPDWORD lpBufferLen
);
```

File

ndascomm.h

Parameters

Parameters	Description
`IN HNDAS hNDASDevice`	Handle to the NDAS Device.
`OUT PBYTE Buffer`	Pointer to a data buffer to store the host address. If NULL, NdasCommGetHostAddress returns required buffer length to lpBufferLen
`IN OUT LPDWORD lpBufferLen`	A size of the address buffer.

Returns

If the function succeeds, the return value is non-zero. If the function fails, the return value is zero. To get extended error information, call GetLastError.

Description

The NdasCommGetHostAddress function retrieves the host address of the NDAS Device. If the connection type is NDASCOMM_TRANSPORT_LPX, the NdasCommGetHostAddress function returns 6 bytes of LPX Address.

2.1.11 NdasCommGetTransmitTimeout

The NdasCommGetTransmitTimeout retrieves the transmit timeout value.

```
NDASCOMM_API BOOL NDASAPICALL NdasCommGetTransmitTimeout(
    IN HNDAS hNDASDevice,
    OUT LPDWORD dwTimeout
);
```

File

ndascomm.h

Parameters

Parameters	Description
`IN HNDAS hNDASDevice`	Handle to the NDAS Device.
`OUT LPDWORD dwTimeout`	Indicator specifying the time-out interval, in milliseconds. If dwTimeout is 0, the function's time-out interval never expires(acts in synchronous mode).

Returns

If the function succeeds, the return value is non-zero. If the function fails, the return value is zero. To get extended error information, call GetLastError.

Description

The NdasCommGetTransmitTimeout retrieves the transmit timeout value.

See Also

NdasCommSetTransmitTimeout (see page 33), NdasCommConnect (see page 19)

2.1.12 NdasCommGetUnitDeviceInfo

The NdasCommGetUnitDeviceInfo retrieves static Unit Device information from the NDAS Device.

```
NDASCOMM_API BOOL NDASAPICALL NdasCommGetUnitDeviceInfo(
    IN HNDAS hNDASDevice,
    OUT PNDASCOMM_UNIT_DEVICE_INFO pUnitInfo
);
```

File

ndascomm.h

Parameters

Parameters	Description
`IN HNDAS hNDASDevice`	Handle to the NDAS Device.
`OUT PNDASCOMM_UNIT_DEVICE_INFO pUnitInfo`	A structure to retrieve static unit device information.

Returns

If the function succeeds, the return value is non-zero. If the function fails, the return value is zero. To get extended error information, call GetLastError.

Description

The NdasCommGetUnitDeviceInfo retrieves static Unit Device information from the NDAS Device. See NDASCOMM_UNIT_DEVICE_INFO for detailed information.

See Also

_NDASCOMM_UNIT_DEVICE_INFO (see page 52), NdasCommGetUnitDeviceStat (see page 28)

2.1.13 NdasCommGetUnitDeviceStat

The NdasCommGetUnitDeviceStat retrieves dynamic Unit Device information from the NDAS Device.

```
NDASCOMM_API BOOL NDASAPICALL NdasCommGetUnitDeviceStat(
    IN CONST NDASCOMM_CONNECTION_INFO* pConnectionInfo,
    OUT PNDASCOMM_UNIT_DEVICE_STAT pUnitDynInfo,
    IN CONST DWORD dwTimeout,
    IN CONST VOID * hint
);
```

File

ndascomm.h

Parameters

Parameters	Description
`IN CONST NDASCOMM_CONNECTION_INFO* pConnectionInfo`	A connection information to a NDAS Device.
`OUT PNDASCOMM_UNIT_DEVICE_STAT pUnitDynInfo`	A structure to retrieve dynamic unit device information.
`IN CONST DWORD dwTimeout`	Indicator specifying the time-out interval, in milliseconds. If dwTimeout is 0, the function's time-out interval never expires.
`IN CONST VOID * hint`	If hint is not NULL, the function tries the host addresses in hint . If hint is NULL, the function tries all the host addresses until the connection established. If protocol is NDASCOMM_TRANSPORT_LPX, hint should be LPSOCKET_ADDRESS_LIST type.

Returns

If the function succeeds, the return value is non-zero. If the function fails, the return value is zero. To get extended error information, call GetLastError.

Description

The NdasCommGetUnitDeviceStat retrieves dynamic Unit Device information from the NDAS Device. Because the function makes new connection to the NDAS Device as discovery mode, the function receives not a HNDAS handle parameter but NDASCOMM_CONNECTION_INFO parameter with NDASCOMM_LOGIN_TYPE_DISCOVER. The NdasCommGetUnitDeviceStat function does not increase RO or RW count. The function disconnects connection to the NDAS Device after retrieved dynamic information. See PNDASCOMM_UNIT_DEVICE_INFO for detailed information.

See Also

_NDASCOMM_CONNECTION_INFO (see page 45), _NDASCOMM_UNIT_DEVICE_INFO (see page 52), NdasCommGetUnitDeviceInfo (see page 27)

Example

```
// retrieve dynamic unit device information

BOOL bResult;
NDASCOMM_CONNECTION_INFO ConnectionInfo;
PNDASCOMM_UNIT_DEVICE_INFO UnitDynInfo;

ZeroMemory(&ConnectionInfo, sizeof(NDASCOMM_CONNECTION_INFO));
ConnectionInfo.address_type = NDASCOMM_CONNECTION_INFO_TYPE_ID_A;
ConnectionInfo.login_type = NDASCOMM_LOGIN_TYPE_DISCOVER;
ConnectionInfo.UnitNo = 0;
ConnectionInfo.bWriteAccess = FALSE; // Not need RW access
ConnectionInfo.ui64OEMCode = NULL;
ConnectionInfo.protocol = NDASCOMM_TRANSPORT_LPX;
memcpy(ConnectionInfo.sDeviceStringId, "ABCDE12345EFGHI67890", 20);
memcpy(ConnectionInfo.sDeviceStringKey, "12345", 5);

bResult = NdasCommGetUnitDeviceStat(&ConnectionInfo, &UnitDynInfo, 0, NULL);
```

2.1.14 NdasCommIdeCommand

The NdasCommIdeCommand sends ide command and receives the result.

```
NDASCOMM_API BOOL NDASAPICALL NdasCommIdeCommand(
    IN HNDAS hNDASDevice,
    IN OUT PNDASCOMM_IDE_REGISTER pIdeRegister,
    IN OUT PBYTE pWriteData,
    IN UINT32 uiWriteDataLen,
    OUT PBYTE pReadData,
    IN UINT32 uiReadDataLen
);
```

File

ndascomm.h

Parameters

Parameters	Description
`IN HNDAS hNDASDevice`	Handle to the NDAS Device.
`IN OUT PNDASCOMM_IDE_REGISTER pIdeRegister`	A structure with ide register and command information.
`IN OUT PBYTE pWriteData`	If a command needs additional data to send to the NDAS Device, the variable is a pointer to a send data buffer. Otherwise, NULL.
`IN UINT32 uiWriteDataLen`	If pWriteData is not NULL, uiWriteDataLen is number of bytes sending by this call.
`OUT PBYTE pReadData`	If a command needs additional data to receive from the NDAS Device, the variable is a pointer to a receive data buffer. Otherwise, NULL.
`IN UINT32 uiReadDataLen`	If pReadData is not NULL, uiReadDataLen is number of bytes receiving by this call.

Returns

If the function succeeds, the return value is non-zero. If the function fails, the return value is zero. To get extended error information, call GetLastError.

Description

The NdasCommIdeCommand sends ide command and receives the result. The function is designed to process all commands including WIN_READ, WIN_WRITE, WIN_VERIFY, etc. Caller MUST set NDASCOMM_IDE_REGISTER structure correctly. Unless, the NdasCommIdeCommand function will fail or halt. Because the NdasCommIdeCommand function does not use handshaked DMA, LBA information, use_dma, use_48 member in NDASCOMM_IDE_REGISTER must be set or reset correctly. See ATA/ATAPI specification document and NDASCOMM_IDE_REGISTER for detailed information.

See Also

_NDASCOMM_IDE_REGISTER (see page 49)

Example

```
// write 1 sector to sector 0 using DMA and LBA 48

BOOL bResult;
NDASCOMM_IDE_REGISTER reg;
BYTE send_buffer[512];
UINT64 Location = 0;
UINT SectorCount = 1;

reg.use_dma = 1;
reg.use_48 = 1;
reg.device.lba_head_nr = 0; // sector : 0
reg.device.dev = 0; // target id : 0
reg.device.lba = 1; // use lba
reg.command.command = 0x35; // == WIN_WRITEDMA_EXT;
reg.reg.named_48.cur.features; // reserved
reg.reg.named_48.prev.features; // reserved
```

```
reg.reg.named_48.cur.sector_count = (BYTE)((SectorCount & 0x00FF) >> 0);
reg.reg.named_48.prev.sector_count = (BYTE)((SectorCount & 0xFF00) >> 8);
reg.reg.named_48.cur.lba_low = (BYTE)((Location & 0x00000000000000FF) >> 0);
reg.reg.named_48.prev.lba_low = (BYTE)((Location & 0x000000000000FF00) >> 8);
reg.reg.named_48.cur.lba_mid = (BYTE)((Location & 0x0000000000FF0000) >> 16);
reg.reg.named_48.prev.lba_mid = (BYTE)((Location & 0x00000000FF000000) >> 24);
reg.reg.named_48.cur.lba_high = (BYTE)((Location & 0x000000FF00000000) >> 32);
reg.reg.named_48.prev.lba_high = (BYTE)((Location & 0x0000FF0000000000) >> 40);

bResult = NdasCommIdeCommand(hNDASDevice, &reg, send_buffer, sizeof(send_buffer), NULL, 0);
```

2.1.15 NdasCommInitialize

The NdasCommInitialize function initiates use of NdasComm.DLL or NdasComm.LIB by a process.

```
NDASCOMM_API BOOL NDASAPICALL NdasCommInitialize();
```

File

ndascomm.h

Returns

The NdasCommInitialize function returns TRUE if successful. Otherwise, it returns FALSE.

Description

The NdasCommInitialize function must be the first NdasComm function called by an application or DLL.

2.1.16 NdasCommSetFeatures

The NdasCommSetFeatures set features of the unit device.

```
NDASCOMM_API BOOL NDASAPICALL NdasCommSetFeatures(
    IN HNDAS hNDASDevice,
    IN BYTE feature,
    IN BYTE param0,
    IN BYTE param1,
    IN BYTE param2,
    IN BYTE param3
);
```

File

ndascomm.h

Parameters

Parameters	Description
IN HNDAS hNDASDevice	Handle to the NDAS Device.
IN BYTE feature	Subcommand code. Features register.
IN BYTE param0	Subcommand specific. Sector Count register.
IN BYTE param1	Subcommand specific. LBA Low register.
IN BYTE param2	Subcommand specific. LBA Mid register.
IN BYTE param3	Subcommand specific. LBA High register.

Returns

If the function succeeds, the return value is non-zero. If the function fails, the return value is zero. To get extended error information, call GetLastError.

Description

The NdasCommSetFeatures function send WIN_SETFEATURES command and parameters to the unit device. See ATA/ATAPI specification document for detailed information.

Example

```
// Set dma of the unit device to Ultra DMA level 5
BOOL bResult;
BYTE feature = 0x03; // Set transfer mode based on value in Sector Count register.
BYTE dma_feature_mode = 0x40 | 5; // Ultra DMA 5
bResult = NdasCommSetFeatures(hNDASDevice, feature, dma_feature_mode, 0, 0, 0);
```

2.1.17 NdasCommSetTransmitTimeout

The NdasCommSetTransmitTimeout sets the transmit timeout value.

```
NDASCOMM_API BOOL NDASAPICALL NdasCommSetTransmitTimeout(
    IN HNDAS hNDASDevice,
    IN CONST DWORD dwTimeout
);
```

File

ndascomm.h

Parameters

Parameters	Description
`IN HNDAS hNDASDevice`	Handle to the NDAS Device.
`IN CONST DWORD dwTimeout`	Indicator specifying the time-out interval, in milliseconds. If dwTimeout is 0, the function's time-out interval never expires(acts in synchronous mode).

Returns

If the function succeeds, the return value is non-zero. If the function fails, the return value is zero. To get extended error information, call GetLastError.

Description

The NdasCommSetTransmitTimeout sets the transmit timeout value.

See Also

NdasCommGetTransmitTimeout (see page 26), NdasCommConnect (see page 19)

2.1.18 NdasCommUninitialize

The NdasCommUninitialize function makes end use of NdasComm.DLL or NdasComm.LIB by a process.

```
NDASCOMM_API BOOL NDASAPICALL NdasCommUninitialize();
```

File

ndascomm.h

Returns

The NdasCommUnnitialize function returns TRUE if successful. Otherwise, it returns FALSE.

Description

The NdasCommUnnitialize function must be the last NdasComm function called by an application or DLL.

2.1.19 NdasCommVendorCommand

The NdasCommVendorCommand sends vendor specific command and receives the results.

```
NDASCOMM_API BOOL NDASAPICALL NdasCommVendorCommand(
    IN HNDAS hNDASDevice,
    IN NDASCOMM_VCMD_COMMAND vop_code,
    IN OUT PNDASCOMM_VCMD_PARAM param,
    IN OUT PBYTE pWriteData,
    IN UINT32 uiWriteDataLen,
    IN OUT PBYTE pReadData,
    IN UINT32 uiReadDataLen
);
```

File

ndascomm.h

Parameters

Parameters	Description
`IN HNDAS hNDASDevice`	Handle to the NDAS Device.
`IN NDASCOMM_VCMD_COMMAND vop_code`	Specifies vendor command code. Some command codes need supervisor permission. See Description for details.
`IN OUT PNDASCOMM_VCMD_PARAM param`	Pointer to parameter for vendor command.
`IN OUT PBYTE pWriteData`	If not NULL, Additional data to send with vendor command. Contents in the buffer will corrupted after sent.
`IN UINT32 uiWriteDataLen`	Length of the write buffer.
`IN OUT PBYTE pReadData`	If not NULL, Additional data to receive with vendor command.
`IN UINT32 uiReadDataLen`	Length of the read buffer.

Returns

If the function succeeds, the return value is non-zero. If the function fails, the return value is zero. To get extended error information, call GetLastError.

Description

The NdasCommVendorCommand sends vendor specific command and receives the results. Some commands need to be executed by supervisor permission.

See NDASCOMM_VCMD_PARAM for detailed commands and parameters.

See Also

_NDASCOMM_VCMD_PARAM (see page 39), NdasCommConnect (see page 19)

Example

```
// Get and Set HDD standby timer
// Make sure that The NDAS Device is connected with supervisor permission.

    NDASCOMM_VCMD_PARAM param_vcmd;

    // read standby timer
    bResult = NdasCommVendorCommand(
        hNDAS,
        ndascomm_vcmd_get_standby_timer,
        &param_vcmd,
        NULL, 0, NULL, 0);

    // set standby timer to 30 minutes
    param_vcmd.SET_STANDBY_TIMER.EnableTimer = TRUE;
    param_vcmd.SET_STANDBY_TIMER.TimeValue = 30;

    bResult = NdasCommVendorCommand(
        hNDAS,
```

```
        ndascomm_vcmd_set_standby_timer,
        &param_vcmd,
        NULL, 0, NULL, 0);

    // read standby timer again
    bResult = NdasCommVendorCommand(
        hNDAS,
        ndascomm_vcmd_get_standby_timer,
        &param_vcmd,
        NULL, 0, NULL, 0);
```

2.2 Structs, Records, Enums

2.2.1 NDASCOMM_BIN_PARAM_TARGET_DATA

```
struct _NDASCOMM_BIN_PARAM_TARGET_DATA {
  UINT8 ParamType;
  UINT8 GetOrSet;
  UINT16 Reserved1;
  UINT32 TargetID;
  UINT64 TargetData;
};
```

File

ndascomm_type.h

Description

Binary type parameter for text request command. Not used. Use NdasCommGetUnitDeviceStat (see page 28) function to retrieve the information instead.

See Also

NdasCommGetUnitDeviceStat (see page 28)

2.2.2 NDASCOMM_VCMD_PARAM

Specifies the attributes of connection information. This structure is used to create connection to a NDAS Device.

```
union _NDASCOMM_VCMD_PARAM {
  struct {
    BYTE Data[12];
  } COMMON;
  struct {
    UINT32 RetTime;
  } SET_RET_TIME;
  struct {
    UINT32 MaxConnTime;
  } SET_MAX_CONN_TIME;
  struct {
    UCHAR SupervisorPassword[8];
  } SET_SUPERVISOR_PW;
  struct {
    UCHAR UserPassword[8];
  } SET_USER_PW;
  struct {
    BOOL EncryptHeader;
    BOOL EncryptData;
  } SET_ENC_OPT;
  struct {
    BOOL EnableTimer;
    UINT32 TimeValue;
  } SET_STANDBY_TIMER;
  struct {
    UINT32 Obsolete;
  } RESET;
  struct {
    UINT32 AddressLPX[6];
  } SET_LPX_ADDRESS;
  struct {
    UINT8 Index;
    UINT32 SemaCounter;
  } SET_SEMA;
  struct {
    UINT8 Index;
    UINT32 SemaCounter;
  } FREE_SEMA;
  struct {
    UINT8 Index;
    UINT32 SemaCounter;
  } GET_SEMA;
  struct {
    UINT8 Index;
    UINT32 AddressLPX[6];
  } GET_OWNER_SEMA;
  struct {
    UINT32 Delay;
  } SET_DELAY;
  struct {
    UINT32 TimeValue;
  } GET_DELAY;
  struct {
    UINT32 Obsolete;
  } SET_DYNAMIC_MAX_CONN_TIME;
  struct {
    UINT32 TimeValue;
  } SET_DYNAMIC_RET_TIME;
  struct {
    UINT32 RetTime;
  } GET_DYNAMIC_RET_TIME;
  struct {
    BOOL EncryptHeader;
```

```
    BOOL EncryptData;
  } SET_D_ENC_OPT;
  struct {
    BOOL EncryptHeader;
    BOOL EncryptData;
  } GET_D_ENC_OPT;
  struct {
    UINT32 RetTime;
  } GET_RET_TIME;
  struct {
    UINT32 MaxConnTime;
  } GET_MAX_CONN_TIME;
  struct {
    BOOL EnableTimer;
    UINT32 TimeValue;
  } GET_STANDBY_TIMER;
};
```

File

ndascomm_type.h

Members

Members	Description
`struct { BYTE Data[12]; } COMMON;`	Reserved, do not use
`struct { UINT32 RetTime; } SET_RET_TIME;`	ndascomm_vcmd_set_ret_time RetTime : [in] Retransmission time to set, in microsecond. Should be greater than 0.
`struct { UINT32 MaxConnTime; } SET_MAX_CONN_TIME;`	ndascomm_vcmd_set_max_conn_time MaxConnTime : [in] Maximum connection time to set, in second. Should be greater than 0.
`struct { UCHAR SupervisorPassword[8]; } SET_SUPERVISOR_PW;`	ndascomm_vcmd_set_supervisor_pw SupervisorPassword : [in] Supervisor password. 8 bytes
`struct { UCHAR UserPassword[8]; } SET_USER_PW;`	IN
`struct { BOOL EncryptHeader; BOOL EncryptData; } SET_ENC_OPT;`	ndascomm_vcmd_set_enc_opt EncryptHeader : [in] TRUE if encrypt header, FALSE otherwise. EncryptData : [in] TRUE if encrypt data, FALSE otherwise.
`struct { BOOL EnableTimer; UINT32 TimeValue; } SET_STANDBY_TIMER;`	IN
`UINT32 TimeValue;`	minutes, < 2^31
`struct { UINT32 Obsolete; } RESET;`	ndascomm_vcmd_reset Do not set any value.
`struct { UINT32 AddressLPX[6]; } SET_LPX_ADDRESS;`	IN
`struct { UINT8 Index; UINT32 SemaCounter; } SET_SEMA;`	IN, OUT
`UINT8 Index;`	IN, 0 ~ 3
`UINT32 SemaCounter;`	OUT
`struct { UINT8 Index; UINT32 SemaCounter; } FREE_SEMA;`	IN, OUT
`UINT8 Index;`	IN, 0 ~ 3
`UINT32 SemaCounter;`	OUT
`struct { UINT8 Index; UINT32 SemaCounter; } GET_SEMA;`	IN, OUT

`UINT8 Index;`	IN, 0 ~ 3
`UINT32 SemaCounter;`	OUT

`struct {`
`UINT8 Index;`
`UINT32 AddressLPX[6];`
`} GET_OWNER_SEMA;`
	IN, OUT

`UINT8 Index;`	IN, 0 ~ 3
`UINT32 AddressLPX[6];`	OUT

`struct {`
`UINT32 Delay;`
`} SET_DELAY;`
	ndascomm_vcmd_set_delay
	Delay : [in] Time delay between packets from the NDAS Device to the host, in nanosecond. Should be multiples of 8. Default is 0

`struct {`
`UINT32 TimeValue;`
`} GET_DELAY;`
	OUT

`UINT32 TimeValue;`	nano seconds, >= 8

`struct {`
`UINT32 Obsolete;`
`} SET_DYNAMIC_MAX_CONN_TIME;`
	obsolete

`struct {`
`UINT32 TimeValue;`
`} SET_DYNAMIC_RET_TIME;`
	IN

`UINT32 TimeValue;`	micro second

`struct {`
`UINT32 RetTime;`
`} GET_DYNAMIC_RET_TIME;`
	OUT

`UINT32 RetTime;`	micro second, >= 1

`struct {`
`BOOL EncryptHeader;`
`BOOL EncryptData;`
`} SET_D_ENC_OPT;`
	IN

`struct {`
`BOOL EncryptHeader;`
`BOOL EncryptData;`
`} GET_D_ENC_OPT;`
	IN, OUT

`struct {`
`UINT32 RetTime;`
`} GET_RET_TIME;`
	OUT

`struct {`
`UINT32 MaxConnTime;`
`} GET_MAX_CONN_TIME;`
	OUT

`UINT32 MaxConnTime;`	second(*)

`struct {`
`BOOL EnableTimer;`
`UINT32 TimeValue;`
`} GET_STANDBY_TIMER;`
	OUT

`UINT32 TimeValue;`	minutes, < 2^31

Description

address_type member can have one of the following values.

Value	Meaning
NDASCOMM_CONNECTION_INFO_TYPE_ADDR_LPX	Use LpxAddress member to address a NDAS Device.
NDASCOMM_CONNECTION_INFO_TYPE_ADDR_IP	Use IP member to address a NDAS Device.
NDASCOMM_CONNECTION_INFO_TYPE_ID_W	Use wszDeviceStringId and wszDeviceStringKey member to address a NDAS Device.
NDASCOMM_CONNECTION_INFO_TYPE_ID_A	Use szDeviceStringId and szDeviceStringKey member to address a NDAS Device.
NDASCOMM_CONNECTION_INFO_TYPE_ID_DEVICE	Use Device ID to address a NDAS Device

login_type member can have one of the following values.

Value	Meaning
NDASCOMM_LOGIN_TYPE_NORMAL	Connects to the NDAS Device to operate IDE Commands. Use this value only.
NDASCOMM_LOGIN_TYPE_DISCOVER	Internal use only. There is no exported API function that supports discover login type.

transport member can have one of the following values.

Value	Meaning
NDASCOMM_TRANSPORT_LPX	Use LPX transport protocol.
NDASCOMM_TRANSPORT_IP	Not supported.

See Also

NdasCommVendorCommand (see page 35)

2.2.3 NDASCOMM_BIN_PARAM_TARGET_LIST

```
struct _NDASCOMM_BIN_PARAM_TARGET_LIST {
  UINT8 ParamType;
  UINT8 NRTarget;
  UINT16 Reserved1;
  NDASCOMM_BIN_PARAM_TARGET_LIST_ELEMENT PerTarget[2];
};
```

File

ndascomm_type.h

Description

Binary type parameter for text request command. Not used. Use NdasCommGetUnitDeviceStat (see page 28) function to retrieve the information instead.

See Also

NdasCommGetUnitDeviceStat (see page 28)

2.2.4 NDASCOMM_BIN_PARAM_TARGET_LIST_ELEMENT

```
struct _NDASCOMM_BIN_PARAM_TARGET_LIST_ELEMENT {
  UINT32 TargetID;
  UINT8 NRRWHost;
  UINT8 NRROHost;
  UINT16 Reserved1;
  UINT32 TargetData0;
  UINT32 TargetData1;
};
```

File

ndascomm_type.h

Description

Binary type parameter for text request command. Not used. Use NdasCommGetUnitDeviceStat (see page 28) function to retrieve the information instead.

See Also

NdasCommGetUnitDeviceStat (see page 28)

2.2.5 NDASCOMM_CONNECTION_INFO

Specifies the attributes of connection information. This structure is used to create connection to a NDAS Device.

```
struct _NDASCOMM_CONNECTION_INFO {
  DWORD address_type;
  DWORD login_type;
  DWORD UnitNo;
  BOOL bWriteAccess;
  BOOL bSupervisor;
  UINT64 ui64OEMCode;
  DWORD protocol;
  union {
    BYTE AddressLPX[6];
    BYTE AddressIP[4];
    BYTE DeviceID[6];
    struct {
      CHAR szDeviceStringId[20 + 1];
      CHAR szDeviceStringKey[5 + 1];
    } DeviceIDA;
    struct {
      WCHAR wszDeviceStringId[20 + 1];
      WCHAR wszDeviceStringKey[5 + 1];
    } DeviceIDW;
  };
};
```

File

ndascomm_type.h

Members

Members	Description
`DWORD address_type;`	[in] Value specifying which address type will be used to make connection to a NDAS Device. See description for available values.
`DWORD login_type;`	[in] Value specifying which login type will be used to make connection to a NDAS Device. See description for available values.
`DWORD UnitNo;`	[in] Value specifying which unit of the NDAS Device will be used. Currently there is only 1 unit attached to a NDAS Device. Set this value to 0.
`BOOL bWriteAccess;`	[in] If true, connects to the NDAS Device with read-write mode. Otherwise, connects to the NDAS Device with read only mode.
`BOOL bSupervisor;`	[in] If true, connects to the NDAS Device with supervisor right. Otherwise, connects to the NDAS Device with normal user mode.
`UINT64 ui64OEMCode;`	[in] Value to verify host as valid user. This value is used as password. If this value is 0, API function tries with default code for normal user. To login as super user, you must specify this value. NdasCommVendorCommand (see page 35) needs super user privilege for some commands.
`DWORD protocol;`	[in] Value specifying which protocol will be used to make a connection to the NDAS Device. See description for available values.
`BYTE AddressLPX[6];`	[in] Used if address_type is NDASCOMM_CONNECTION_INFO_TYPE_ADDR_LPX. LPX address of the NDAS Device.
`BYTE AddressIP[4];`	[in] Used if address_type is NDASCOMM_CONNECTION_INFO_TYPE_ADDR_IP. Not supported.
`BYTE DeviceID[6];`	[in] Used if address_type is NDASCOMM_CONNECTION_INFO_TYPE_ID_DEVICE.
`CHAR szDeviceStringId[20 + 1];`	[in] Used if address_type is NDASCOMM_CONNECTION_INFO_TYPE_ID_A. 20 characters of case insensitive ANSI code.
`CHAR szDeviceStringKey[5 + 1];`	[in] Used if address_type is NDASCOMM_CONNECTION_INFO_TYPE_ID_A and bWriteAccess is not 0. 5 characters of case insensitive ANSI code.
`WCHAR wszDeviceStringId[20 + 1];`	[in] Used if address_type is NDASCOMM_CONNECTION_INFO_TYPE_ID_W. 20 characters of case insensitive UNICODE.
`WCHAR wszDeviceStringKey[5 + 1];`	[in] Used if address_type is NDASCOMM_CONNECTION_INFO_TYPE_ID_W and bWriteAccess is not 0. 5 characters of case insensitive UNICODE.

Description

address_type member can have one of the following values.

Value	Meaning
NDASCOMM_CONNECTION_INFO_TYPE_ADDR_LPX	Use LpxAddress member to address a NDAS Device.
NDASCOMM_CONNECTION_INFO_TYPE_ADDR_IP	Use IP member to address a NDAS Device.
NDASCOMM_CONNECTION_INFO_TYPE_ID_W	Use wszDeviceStringId and wszDeviceStringKey member to address a NDAS Device.
NDASCOMM_CONNECTION_INFO_TYPE_ID_A	Use szDeviceStringId and szDeviceStringKey member to address a NDAS Device.
NDASCOMM_CONNECTION_INFO_TYPE_ID_DEVICE	Use Device ID to address a NDAS Device

login_type member can have one of the following values.

Value	Meaning
NDASCOMM_LOGIN_TYPE_NORMAL	Connects to the NDAS Device to operate IDE Commands. Use this value only.
NDASCOMM_LOGIN_TYPE_DISCOVER	Internal use only. There is no exported API function that supports discover login type.

protocol member can have one of the following values.

Value	Meaning
NDASCOMM_TRANSPORT_LPX	Use LPX transport protocol.
NDASCOMM_TRANSPORT_IP	Not supported.

See Also

NdasCommConnect (see page 19), NdasCommGetUnitDeviceStat (see page 28)

2.2.6 NDASCOMM_HANDLE_INFO_TYPE

```
enum _NDASCOMM_HANDLE_INFO_TYPE {
  ndascomm_handle_info_hw_type = 0,
  ndascomm_handle_info_hw_version,
  ndascomm_handle_info_hw_proto_type,
  ndascomm_handle_info_hw_proto_version,
  ndascomm_handle_info_hw_num_slot,
  ndascomm_handle_info_hw_max_blocks,
  ndascomm_handle_info_hw_max_targets,
  ndascomm_handle_info_hw_max_lus,
  ndascomm_handle_info_hw_header_encrypt_algo,
  ndascomm_handle_info_hw_data_encrypt_algo
};
```

File

ndascomm_type.h

Description

Specifies the information type to retrieve from the handle to the NDAS Device.

Value	Meaning
ndascomm_handle_info_hw_type	Hardware type of the NDAS Device. UINT8. This type is not supported.
ndascomm_handle_info_hw_version	Hardware version of the NDAS Device. UINT8. 0 : Ver 1.0 1 : Ver 1.1 2 : Ver 2.0
ndascomm_handle_info_hw_proto_type	Protocol type of the NDAS Device. UINT8. This type is not supported.
ndascomm_handle_info_hw_proto_version	Protocol version of the NDAS Device. UINT8 0 : Ver 1.0 1 : Ver 1.1
ndascomm_handle_info_hw_num_slot	Number of slots attached to the NDAS Device. UINT32
ndascomm_handle_info_hw_max_blocks	Value specifying maximum request blocks. requestable. UINT32
ndascomm_handle_info_hw_max_targets	Maximum number of slots attachable to the NDAS Device. UINT32
ndascomm_handle_info_hw_max_lus	Maximum number of LUs. Not supported. UINT32
ndascomm_handle_info_hw_header_encrypt_algo	Packet header encryption type. UINT16 0: Do not encrypt 1: Encrypt

ndascomm_handle_info_hw_data_encrypt_algo	Packet data encryption type. UINT16 0: Do not encrypt 1: Encrypt

See Also

NdasCommGetDeviceInfo (see page 24)

2.2.7 NDASCOMM_IDE_REGISTER

Specifies and receives IDE command register information to the unit device attached the NDAS Device. Fill registers according to command member. reg, device and command members are bi-directional. So, those will be filled by return register informations.

```
struct _NDASCOMM_IDE_REGISTER {
  UINT8 use_dma;
  UINT8 use_48;
  union {
    UINT8 device;
    struct {
      UINT8 lba_head_nr : 4;
      UINT8 dev : 1;
      UINT8 obs1 : 1;
      UINT8 lba : 1;
      UINT8 obs2 : 1;
    }
  } device;
  union {
    UINT8 command;
    struct {
      UINT8 err : 1;
      UINT8 obs1 : 1;
      UINT8 obs2 : 1;
      UINT8 drq : 1;
      UINT8 obs : 1;
      UINT8 df : 1;
      UINT8 drdy : 1;
      UINT8 bsy : 1;
    } status;
  } command;
  union {
    struct basic@1 basic;
public
    struct {
      UINT8 reg[5];
    }
    struct {
      UINT8 features;
      UINT8 sector_count;
      UINT8 lba_low;
      UINT8 lba_mid;
      UINT8 lba_high;
    } named;
    struct {
      UINT8 reg_prev[5];
      UINT8 reg_cur[5];
    } basic_48;
    struct {
      struct prev@1 prev;
public
      struct {
        UINT8 features;
        UINT8 sector_count;
        UINT8 lba_low;
        UINT8 lba_mid;
        UINT8 lba_high;
      }
      struct {
        UINT8 features;
        UINT8 sector_count;
        UINT8 lba_low;
        UINT8 lba_mid;
        UINT8 lba_high;
      } cur;
    } named_48;
```

```
      struct {
        UINT8 obs1[5];
        union {
          UINT8 err_na;
          struct {
            UINT8 obs : 1;
            UINT8 nm : 1;
            UINT8 abrt : 1;
            UINT8 mcr : 1;
            UINT8 idnf : 1;
            UINT8 mc : 1;
            UINT8 wp : 1;
            UINT8 icrc : 1;
          } err_op;
        } err;
        UINT8 obs2[4];
      } ret;
    } reg;
  };
```

File

ndascomm_type.h

Members

Members	Description
`UINT8 lba_head_nr : 4;`	reserved(used as 4 MSBs)
`UINT8 dev : 1;`	set with target id
`UINT8 obs1 : 1;`	obsolete
`UINT8 lba : 1;`	1 : use lba
`UINT8 obs2 : 1;`	obsolete
`UINT8 err : 1;`	ERR (only valid bit)
`UINT8 obs1 : 1;`	obsolete
`UINT8 obs2 : 1;`	obsolete
`UINT8 drq : 1;`	DRQ
`UINT8 obs : 1;`	obsolete
`UINT8 df : 1;`	DF
`UINT8 drdy : 1;`	DRDY
`UINT8 bsy : 1;`	BSY

Description

[Members]

use_dma

[in] 1 : DMA, 0 : PIO

use_48

[in] 1 : use 48 bit address feature, 0 : does not use. Setting LBA 48 mode is required for some IDE commands.

device

[in, out] Device register

command

[in, out] Command(in), Status(out) register

command.command

[in] Command register

command.status

[out] Status register

reg

[in, out] The reg member is union of 5 types of structure. Select structure by purpose.

reg.basic, reg.named

[in, out] Used when LBA 48 mode is not set.

reg.basic_48, reg.named_48

[in, out] Used when LBA 48 mode is set. The NDAS Device writes prev member first, cur member next to the unit device.

reg.ret.err.err_op

[out] Valid if command.status.err is set.

See Also

NdasCommIdeCommand (see page 29)

2.2.8 NDASCOMM_UNIT_DEVICE_INFO

Receives the static attributes of unit device attached to the NDAS Device. All attributes are obtained by WIN_IDENTIFY or WIN_PIDENTIFY IDE command. Using NdasCommIdeCommand (see page 29), same information can be obtained also.

```
struct _NDASCOMM_UNIT_DEVICE_INFO {
  UINT64 SectorCount;
  BOOL bLBA;
  BOOL bLBA48;
  BOOL bPIO;
  BOOL bDma;
  BOOL bUDma;
  BYTE Model[40];
  BYTE FwRev[8];
  BYTE SerialNo[20];
  WORD MediaType;
};
```

File

ndascomm_type.h

Members

Members	Description
`BOOL bLBA;`	Non zero if the unit device supports LBA mode.
`BOOL bLBA48;`	Non zero if the unit device supports LBA 48 mode.
`BOOL bPIO;`	Non zero if the unit device supports PIO mode.
`BOOL bDma;`	Non zero if the unit device supports DMA mode.
`BOOL bUDma;`	Non zero if the unit device supports Ultra DMA mode.
`BYTE Model[40];`	Model number(40 ASCII characters)
`BYTE FwRev[8];`	Firmware revision(8 ASCII characters)
`BYTE SerialNo[20];`	Serial number(20 ASCII characters)
`WORD MediaType;`	Media type of the unit device. See description for available values.

Description

MediaType member can have one of the following values.

Value	Meaning
NDAS_UNITDEVICE_MEDIA_TYPE_UNKNOWN_DEVICE	Unknown(not supported)
NDAS_UNITDEVICE_MEDIA_TYPE_BLOCK_DEVICE	Non-packet mass-storage device (HDD)
NDAS_UNITDEVICE_MEDIA_TYPE_COMPACT_BLOCK_DEVICE	Non-packet compact storage device (Flash card)
NDAS_UNITDEVICE_MEDIA_TYPE_CDROM_DEVICE CD-ROM	device (CD/DVD)
NDAS_UNITDEVICE_MEDIA_TYPE_OPMEM_DEVICE	Optical memory device (MO)

See Also

NdasCommGetUnitDeviceInfo (see page 27), NdasCommIdeCommand (see page 29)

2.2.9 NDASCOMM_UNIT_DEVICE_STAT

Receives the dynamic attributes of unit device attached to the NDAS Device.

```
struct _NDASCOMM_UNIT_DEVICE_STAT {
  UINT32 iNRTargets;
  BOOL bPresent;
  UINT32 NRRWHost;
  UINT32 NRROHost;
  UINT64 TargetData;
};
```

File

ndascomm_type.h

Members

Members	Description
`UINT32 iNRTargets;`	Number of unit device(s) which is attached to the NDAS Device.
`BOOL bPresent;`	TRUE if present, FALSE otherwise
`UINT32 NRRWHost;`	Number of hosts count which is connected to the NDAS Device with read write privilage.
`UINT32 NRROHost;`	Number of hosts count which is connected to the NDAS Device with read only privilage.
`UINT64 TargetData;`	Reserved

Description

Receives the dynamic attributes of unit device attached to the NDAS Device.

See Also

NdasCommGetUnitDeviceStat (see page 28)

2.3 Types

2.3.1 NDASCOMM_VCMD_COMMAND

```
typedef enum _NDASCOMM_VCMD_COMMAND NDASCOMM_VCMD_COMMAND;
```

File

ndascomm_type.h

Description

Specifies the vendor command type parameter for NdasCommVendorCommand (see page 35)

Value	Meaning	Permission	Min Ver
ndascomm_vcmd_set_ret_time	Sets retransmission time delay when the transmission from the NDAS Device to the host has failed.	supervisor	1.0
ndascomm_vcmd_set_max_conn_time	Sets maximum connection time. If no packet was sent to the NDAS Device for the time, connection would be closed.	supervisor	1.0
ndascomm_vcmd_set_supervisor_pw	Sets 8 bytes of supervisor password of the NDAS Device.	supervisor	1.0
ndascomm_vcmd_set_user_pw	Sets 8 bytes of normal user password of the NDAS Device.	supervisor	1.0
ndascomm_vcmd_set_enc_opt	Sets header and data encrypt option for the NDAS Device.	supervisor	1.1
ndascomm_vcmd_set_standby_timer	Enables/disables standby timer and set timer value for the NDAS Device. the NDAS Device sends WIN_STANDBY1 ATA command to the attached unit device(s).	supervisor	1.1
ndascomm_vcmd_reset	Resets the NDAS Device. The NDAS Device will reboot as soon as the command is accepted.	supervisor	1.0
ndascomm_vcmd_set_lpx_address	not implemented		1.1
ndascomm_vcmd_set_sema	Sets a selected semaphore of the NDAS Device and retrieves auto-increasing semaphore counter.	user	1.1
ndascomm_vcmd_free_sema	Frees selected semaphore of the NDAS Device.	user	1.1
ndascomm_vcmd_get_sema	Retrieves a selected semaphore counter of the NDAS Device.	user	1.1
ndascomm_vcmd_get_owner_sema	Retrieves LPX address of the host which locked selected semaphore of the NDAS Device.	user	1.1
ndascomm_vcmd_set_delay	Sets the maximum delay time between each packet from the NDAS Device to host.	user	2.0
ndascomm_vcmd_get_delay	Retrieves the maximum delay time between each packet from the NDAS Device to host.	user	2.0
ndascomm_vcmd_set_dynamic_max_conn_time	obsolete		
ndascomm_vcmd_set_dynamic_ret_time	Sets retransmission time delay when the transmission from the NDAS Device to the host has failed. This value is not saved to the NDAS Device and applied to the connection only.	user	1.1
ndascomm_vcmd_get_dynamic_ret_time	Retrieves retransmission time delay when the transmission from the NDAS Device to the host has failed. This value is not saved to the NDAS Device and applied to the connection only.	user	1.1

ndascomm_vcmd_set_d_enc_opt	not implemented		
ndascomm_vcmd_get_d_enc_opt	not implemented		
ndascomm_vcmd_get_ret_time	Retrieves retransmission time delay when the transmission from the NDAS Device to the host has failed.	both	1.1
ndascomm_vcmd_get_max_conn_time	Retrieves maximum connection time. If no packet was sent to the NDAS Device for the time, connection would be closed.	both	1.1
ndascomm_vcmd_get_standby_timer	Retrieves standby timer setting for the NDAS Device. the NDAS Device sends WIN_STANDBY1 ATA command to the attached unit device(s).	both	1.1

2.3.2 NDASCOMM_HANDLE_INFO_TYPE

```
typedef enum _NDASCOMM_HANDLE_INFO_TYPE {
  ndascomm_handle_info_hw_type = 0,
  ndascomm_handle_info_hw_version,
  ndascomm_handle_info_hw_proto_type,
  ndascomm_handle_info_hw_proto_version,
  ndascomm_handle_info_hw_num_slot,
  ndascomm_handle_info_hw_max_blocks,
  ndascomm_handle_info_hw_max_targets,
  ndascomm_handle_info_hw_max_lus,
  ndascomm_handle_info_hw_header_encrypt_algo,
  ndascomm_handle_info_hw_data_encrypt_algo
} NDASCOMM_HANDLE_INFO_TYPE;
```

File

ndascomm_type.h

Description

Specifies the information type to retrieve from the handle to the NDAS Device.

Value	Meaning
ndascomm_handle_info_hw_type	Hardware type of the NDAS Device. UINT8. This type is not supported.
ndascomm_handle_info_hw_version	Hardware version of the NDAS Device. UINT8. 0 : Ver 1.0 1 : Ver 1.1 2 : Ver 2.0
ndascomm_handle_info_hw_proto_type	Protocol type of the NDAS Device. UINT8. This type is not supported.
ndascomm_handle_info_hw_proto_version	Protocol version of the NDAS Device. UINT8 0 : Ver 1.0 1 : Ver 1.1
ndascomm_handle_info_hw_num_slot	Number of slots attached to the NDAS Device. UINT32
ndascomm_handle_info_hw_max_blocks	Value specifying maximum request blocks. requestable. UINT32
ndascomm_handle_info_hw_max_targets	Maximum number of slots attachable to the NDAS Device. UINT32
ndascomm_handle_info_hw_max_lus	Maximum number of LUs. Not supported. UINT32
ndascomm_handle_info_hw_header_encrypt_algo	Packet header encryption type. UINT16 0: Do not encrypt 1: Encrypt

ndascomm_handle_info_hw_data_encrypt_algo	Packet data encryption type. UINT16 0: Do not encrypt 1: Encrypt

See Also

NdasCommGetDeviceInfo (see page 24)

Index

A

About NDASComm 9

L

LPX Protocol 7

N

NDAS API Architecture 2
NDAS Device 3
NDAS Device Version 4
NDAS Unit Device 6
NDASCOMM_BIN_PARAM_TARGET_DATA 38
NDASCOMM_BIN_PARAM_TARGET_LIST 43
NDASCOMM_BIN_PARAM_TARGET_LIST_ELEMENT 44
NDASCOMM_CONNECTION_INFO 45
NDASCOMM_HANDLE_INFO_TYPE 47, 57
NDASCOMM_IDE_REGISTER 49
NDASCOMM_UNIT_DEVICE_INFO 52
NDASCOMM_UNIT_DEVICE_STAT 53
NDASCOMM_VCMD_COMMAND 55
NDASCOMM_VCMD_PARAM 39
NdasCommBlockDeviceRead 15
NdasCommBlockDeviceVerify 16
NdasCommBlockDeviceWrite 17
NdasCommBlockDeviceWriteSafeBuffer 18
NdasCommConnect 19
NdasCommDisconnect 21
NdasCommGetAPIVersion 22
NdasCommGetDeviceID 23
NdasCommGetDeviceInfo 24
NdasCommGetHostAddress 25
NdasCommGetTransmitTimeout 26
NdasCommGetUnitDeviceInfo 27
NdasCommGetUnitDeviceStat 28
NdasCommIdeCommand 29
NdasCommInitialize 31
NdasCommSetFeatures 32
NdasCommSetTransmitTimeout 33
NdasCommUninitialize 34
NdasCommVendorCommand 35

O

Overview 1

S

Symbol Reference 13

U

Using NDASComm 10